On-line Fault Detection and End-of-Batch Quality Prediction for Batch Processes Incorporating On-line Synchronisation and Phase Identification

by

Travis Louis Myburgh

Thesis presented in partial fulfilment
of the requirements for the Degree

of

MASTER OF ENGINEERING
(CHEMICAL ENGINEERING)

in the Faculty of Engineering
at Stellenbosch University

Supervisor
Dr L. Auret

Co-Supervisor
Prof Andries J. Burger

December 2016

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

December 2016

Abstract

Batch processes are transient processes, which present a unique monitoring challenge. Whereas the expected output variables of a continuous process centre on specific target values that represent some steady state, batch process variables inherently change from an initial state to a final state. Abnormal events, or faults, that occur in batch processes can be identified with an appropriate monitoring scheme.

Bilinear statistical modelling based on latent methods, such as Partial Least Squares (PLS), have been shown by Nomikos and MacGregor (1995) to be an effective way to detect faults and predict the end-of-batch quality. A batch process monitoring toolbox has previously been developed, which supports off-line modelling and monitoring of batch processes. The limitation of off-line monitoring is that product end-of-batch quality and fault detection can only be assessed once the batch has been completed. On-line fault detection and end-of-batch quality prediction in near-real time endeavors to address this limitation.

The aim of this project was to implement an on-line monitoring platform which can detect faults and predict end-of-batch quality in a timely manner. Synchronization of the batch trajectories is a computationally expensive step in off-line monitoring, and the ability of the on-line platform to produce computationally efficient results comparable to off-line synchronization was assessed. The Relaxed Greedy Time Warping (RGTW) approach (González-Martínez et al., 2011) was used to synchronise the trajectories in an on-line fashion. The approach was able to produce comparable results in a computationally efficient way, but some inaccuracies were observed during flat regions of the batch trajectories.

The nature of phases for accurate modelling of batch data was also investigated. Models were built for batch data partitioned in three ways: no partitions, partitions based on known process stages and partitions based on linear correlation structure between the variables and the end-of-batch quality prediction accuracy was compared. The multiphase partial least squares (MPPLS) algorithm (Camacho, et al., 2007) was used to find the changes in linear correlation among the process variables and partition the data accordingly. Partitioning the data using the MPPLS algorithm showed comparable or statistically significant reductions in the overall end-of-batch prediction error compared with the other two data partitioning methods.

The models were applied on-line to two case studies using the on-line monitoring platform and the fault detection and end-of-batch quality prediction accuracy were assessed for the three data partioning methods. The end-of-batch predictions made using the MPPLS algorithm provided the most accurate end-of-batch predictions. Improper on-line synchronisation caused false alarms in certain areas of the batch trajectory, but faults were detected with relative accuracy.

Opsomming

Enkelladingprosesse is oorgangsprosesse, wat ’n unieke moniteringsuitdaging bied. Terwyl die verwagte uitsetveranderlikes van ’n gestadigde proses gevestig is op spesifieke teikenwaardes wat een of ander ewewigstoestand verteenwoordig, verander enkelladingprosesveranderlikes inherent vanaf ’n aanvanklike toestand na ’n finale toestand. Abnormale gebeure, of foute, kan in enkelladingprosesse geïdentifiseer word met behulp van ’n toepaslike moniteringskema.

Bi-lineêre statistiese modellering gebaseer op latente metodes, soos parsiële kleinste kwadrate (PKK), is deur Nomikos en MacGregor (1995) aangetoon om ’n doeltreffende wyse te wees om foute op te spoor en die kwaliteit met enkellading-einde te voorspel. ’n Enkelladingprosesmonitering-gereedskapkas, wat aflyn modellering en monitering van enkelladingprosesse steun, is reeds ontwikkel. Die beperking van aflyn monitering is dat die produk se enkellading-einde-kwaliteit en foutopsporing slegs geassesseer kan word nadat die enkellading proses voltooi is. Aanlyn foutopsporing en enkellading-einde-kwaliteitvoorspelling in ’n bykans intydse sin poog om hierdie beperking aan te spreek.

Die doel van hierdie projek was om ’n aanlyn moniteringsplatform te implementeer wat foute kan opspoor en kwaliteit met enkellading-einde op ’n tydige wyse kan voorspel. Sinkronisasie van die enkellading-trajekte is ’n rekenaarmatig duur stap in aflyn monitering, en die vermoë van die aanlyn platform om rekenaarmatig doeltreffende resultate te lewer, in vergelyking met aflyn sinkronisasie, is geassesseer. Die ontspanne-gierige-tydkromtrekking-benadering (OGTB-benadering) (González-Martínez et al., 2011) is gebruik om die trajekte op ’n aanlyn wyse te sinkroniseer. Die benadering was in staat om vergelykbare resultate in ’n rekenaarmatig doeltreffende wyse te genereer, maar enkele onakkuraathede is in plat areas van die enkellading-trajekte waargeneem.

Die aard van fases vir akkurate modellering van enkelladingdata is ook ondersoek. Modelle vir enkelladingdata is gebou en op drie maniere verdeel: geen verdelings nie, verdelings gebaseer op bekendeprosesfases, en verdelings gebaseer op lineêrekorrelasiestruktuur tussen die veranderlikes en die enkellading-einde-gehaltevoorspellingsakkuraatheid is vergelyk. Die multifase- parsiële-kleinste-kwadrate-algoritme (MFPKK-algoritme) (Camacho, et al., 2007) is gebruik om die veranderinge in lineêre korrelasie onder die prosesveranderlikes op te spoor, en die data dienooreenkomstig te verdeel. Verdeling van die data, deur gebruik te maak van die MFPKK-algoritme, het vergelykbare of statisties beduidende verminderings in die algehele enkellading-einde-voorspellingsfout getoon wanneer dit met die ander twee dataverdelingsmetodes vergelyk word.

Die modelle is aanlyn toegepas in twee gevallestudies deur die aanlyn moniteringsplatform, en die foutopsporing en akkuraatheid van enkellading-einde-kwaliteitvoorspelling is geassesseer vir die drie dataverdelingsmetodes. Die enkellading-einde-voorspellings wat met behulp van die MFPKK-algoritme gemaak is, het die akkuraatste enkellading-einde-voorspellings verskaf. Ongepaste aanlyn sinkronisasie het valse alarms in sekere areas van die enkellading-trajekte veroorsaak, maar foute is met relatiewe akkuraatheid opspoor.

Acknowledgments

I would like to express my undying gratitude for the persons who help me through the course of this project:

- To my supervisors, Prof A.J. Burger and Dr L. Auret, thank you for your patience and guidance through this journey. It would not have been possible without your motivation.

- To Dr J.P. Barnard, for all the technical guidance and support.

- To my mother Eloise and father Louis, thank you so much for being there for me in every way possible. There were so many tough times, and you always supported me.

- To my brother Chad, I am so grateful to have your support, especially near the end of this project. I don't think this would have been possible without you.

Table of Contents

Nomenclature

Acronym	Description
ANOVA	Analysis of Variance
COW	Correlation Optimised Warping
DFW	Degrees of Freedom Within Groups
DTW	Dynamic Time Warping
FAR	False Alarm Rate
HDDTW	Hybrid Derivative Dynamic Time Warping
IV	Indicator Variable
KDR	Known Data Regression
LOOCV	Leave-One-Out Cross Validation
LMV	Lagged Measurement Vector
LSD	Least Significant Difference
MPCA	Multiway Principal Component Analysis
MPLS	Multiway Partial Least Squares
MPPLS	Multiphase Partial Least Squares
MSE	Mean Squared Error
NIPALS	Non Iterative Linear Partial Least Squares
NOC	Normal Operating Conditions
PC	Principal Component
PCA	Principal Component Analysis
PCC	Pearson's Correlation Coefficient
PLS	Partial Least Squares
PRESS	Predicted Residual Sum of Squares
QPE	Quadratic Prediction Error
QRE	Quadratic Residual Error
RGTW	Relaxed Greedy Time Warping
SIMPLS	Simple Partial Least Squares
SP	Singular Points
SPE	Squared Prediction Error
TSR	Trimmed Score Regression

Symbol	Description
a	Individual principal component
a_{ini}	Initial number of PCs in MPPLS
A	Number of principal components
$c(k)$	Coordinate of points on synchronisation grid
$d(i(k), j(k))$	Local distance between two path points on synchronisation grid
$D(K_{ref}, K_i)$	Cumulated distance
$\boldsymbol{E}$	Matrix of **X** residuals in PCA and PLS
$\boldsymbol{f}_{i(k)}$	Column vector of optimal path points at the k-th point on the grid
$\boldsymbol{F}$	Matrix of **Y** residuals in PLS
$\boldsymbol{F^*}$	Sequence of points on the synchronisation grid
$F_{(A,I-A)_\alpha}$	Critical value of F-statistic at
g	Variable in Chi Squared approximation
h	Variable in Chi Squared approximation
I	Number of batches
I	Individual batch
J	Number of measured process variables
J	Individual measured process variable variable
K	Number of time indexes
K	Individual time index
$l(\boldsymbol{F})$	Lower search space limit of the path $\boldsymbol{F}$
M	Number of product quality variables
M	Individual product quality variable
$minL$	Minimum number of time intervals per phase in MPPLS
P	Number of phases used in modelling
$\boldsymbol{P}$	Projection matrix of **X** for PCA or PLS
P	Individual modelled phase
$\boldsymbol{p_a}$	Eigenvector of the a-th principal component
$\boldsymbol{Q}$	Projection matrix of **Y** for PLS
$\boldsymbol{R}$	SIMPLS weight matrix
s	Variance
s_{t_a}	Estimated variance of latent variable a

SPE_x	Squared Prediction Error statistic
$\boldsymbol{t_a}$	Score of the a-th principal component
$\hat{\boldsymbol{t}}$	Estimated scores vector
$t_{\alpha,df}$	Critical value of t-statistic
${T_A}^2$	Hotelling's T^2 statistic
$\boldsymbol{T}$	Scores matrix for PCA or PLS
$u(\boldsymbol{F})$	Upper search space limit of the path $\boldsymbol{F}$
$\boldsymbol{W}$	NIPALS weight matrix
x_{ijk}	Individual Measured Variable
$\hat{x}$	Reconstructed value of x
$\boldsymbol{x}$	Vector of measured process variables
$\boldsymbol{x_{new}}$	Vector of new measurements
$\boldsymbol{X}$	Two-way matrix of measured process variables
$\boldsymbol{X_{BW}}$	Batch-wise unfolded process variable matrix
$\boldsymbol{X_{ref}}$	Reference trajectory
$\boldsymbol{X_{VW}}$	Variable-wise unfolded process variable matrix
$\tilde{\boldsymbol{X}}$	Unscaled process variable matrix
$\underline{\boldsymbol{X}}$	Three-way matrix of measured process variables
$\boldsymbol{X}^*$	Estimated covariance matrix
$\hat{\boldsymbol{y}}_{\boldsymbol{new}}$	Prediction of quality variable for new batch
$\boldsymbol{Y}$	Matrix of end-of-batch quality variables
$\tilde{\boldsymbol{Y}}$	Unscaled product quality matrix

Greek Symbol	**Description**
A	Significance level
γ	RGTW warping window size
δ	Similarity index
μ	Mean
Π	Number of user defined phases for preliminary phase division
π	Individual phase

List of Figures

List of Tables

CHAPTER 1 Introduction

1.1.1 The Relevance of Batch Processes in Industry

Most industrial processes focus on manufacturing a number of products for use in further downstream processes, or to be sold to a consumer. Generally, the product(s) should meet specific quality outcomes, as products of inferior quality may not be able to be used going forward. It therefore is advantageous to be able to monitor the progress of the process and detect any abnormal events that could result in the product quality outcomes not being met, and intervene if necessary.

In industry, continuous processes are used to manufacture, produce, or process specific material without interruption (Wisner & Stanley, 2008). However, if there are interruptions, the process may have to be stopped and restarted. The processes are not designed to run at these start up and shut down conditions, thus the products are usually not of sufficient quality and must be reprocessed or disposed of. A distinction can also be made between planned interruptions and unplanned interruptions. An example of a planned interruption is scheduled maintenance. An example of an unplanned interruption is contaminants in the process. The latter will be discussed with an example.

In biological processes such as bioethanol production, contamination is a problem (Beckner, et al., 2011). Bacteria such as the *Lactobacillus* species can lead to stuck fermentations and cause a shutdown of the system. Contamination during continuous fermentation is a more serious problem (Chandel, et al., 2007). Bacteria can regrow from a small number of cells, so all the equipment must be cleaned rigorously to get rid of the bacteria. All the mass that crosses the boundary can get contaminated and the products will not be of sufficient quality. Furthermore, the products made during the start-up of the process may also not be of sufficient quality.

Using a batch process in this instance can reduce the effect of these problems as each new batch is started with a new feed. It follows that the process variable trajectories are only dependent on the conditions of the specific run. If a batch process is found to contaminated, only the batch runs that came into contact with the contamination will be affected.

Another potential issue with continuous processes in bioethanol production is the difficulty in maintaining high cell concentration in the fermenter. Using immobilised cells in continuous processes can circumvent this difficulty (Chandel, et al., 2007), but the economic feasibility of changing to a continuous process is still not clear.

The feasibility of using a batch process versus a continuous process is ultimately an economical one. In smaller operations, the advantages of continuous processes may not have as much value as in larger operations. Furthermore, fluctuating market demand and unsteady raw material supply may result in unplanned interruptions in continuous processes, and batch processes may be preferred.

1.1.2 Monitoring of Batch versus Continuous Processes

Over the duration of a batch run, the products are not removed at the rate at which the reactants enter the vessel. Some of the process variables must change with time and thus the process operates under transient conditions. The goal of a batch process is to produce products of a certain quality, hence a successful batch run can be defined as a batch run where the product(s) at the end of the batch is of acceptable quality.

The inherent differences between batch and continuous processes require different approaches to the monitoring procedures followed. The goal of a continuous process is to produce a product with no breaks in time. Generally, continuous processes are controlled by maintaining the process variables close to specific operating points, where the behaviour of the process is approximately linear (Camacho, et al., 2007).

In contrast, batch processes are transient and batch process variables follow trajectories from the beginning to the end of the batch run. Successful batch processes finish once the specified end quality is reached, but this does not always occur at the same time during the process. Small variations in process conditions may cause the batch to proceed at a different rate during all or parts of the process. The goal of batch process monitoring is to assess and maintain these desired trajectories. This may be applied to any process where the process variables change over time in a transient manner. Consequently, batch process monitoring can be extended to incorporate semi-batch processes as well as the transient states of continuous processes (e.g. start-ups and shut-downs). Garcia-Alvarez, Fuente, & Sainz (2012) applied batch process monitoring techniques to detect faults in the evaporation stage of a sugar factory simulation and a laboratory plant.

1.2 Introduction to Modelling and Monitoring of Batch Processes

Modelling of batch processes can be separated into two categories:

1) Fundamental models can be constructed from detailed engineering knowledge about the batch process.
2) Models can be constructed from input-output data recorded in a process

Fundamental models rely heavily on the accuracy of estimated parameters and the engineering knowledge used. While such models may be able to simulate the process reasonably well - the assumptions made when constructing the models may not always apply. A review of fundamental model-based fault detection and isolation (FDI) concepts by Frank (1990) concluded that the FDI approach is problematic where only poor or imprecise models are available, using chemical plants as an example. Other variables that the fundamental models don't account for, such as measurements on mechanical or electrical components, may also have an effect on the performance of the process (Kourti, 2002). The need for precise fundamental modelling is a barrier for using these approaches in batch process monitoring, although they still have value in some applications.

Input-output data-based models are constructed from historical data that describe the process. When using these methods, it is quite difficult to use the data without expert knowledge of the process because the data sets are usually large, with many correlated variables, lots of noise and missing values (Ündey & Çinar, 2002). Monitoring each individual measured variable separately, or univariate process monitoring, is inadequate as the dependence of the measured variables on one another is not taken into account. It is far more useful to analyse the observations of measured variables simultaneously. Multivariate Statistical Analysis Methods based on the Projection to Latent Structures are frequently used for generation of models from input-output data (Nomikos & MacGregor, 1994).

Once a model has been constructed from historical data, or trained, the process can be monitored by applying the model to the incoming data. Statistics generated from this model application can be compared to predefined control limits, and if these statistics exceed these limits an alarm is recorded. Generally, faults in the process cause these statistics to exceed these limits. Accurate fault detection is assessed by determining if these alarms indicate an actual fault in the process. The statistics used for fault detection in this work are discussed in section 2.2.3.

End-of-batch quality can also be predicted if a regression model is trained. During training, a predictor ($\boldsymbol{\beta}$) is calculated that tries to relate the input data to the end-of-batch quality. Applying this predictor to the incoming data can generate predictions about the end-of-batch quality. These predictions are only meaningful when the model can reasonably explain the data, therefore can only be applied if no faults occur in the process. The equations concerning end-of-batch prediction are given in section 2.2.2.

1.2.1 Off-line versus On-line Monitoring

When comparing data, it is advantageous to have the whole batch trajectory available for analysis. For these data to be available, the batch run must have been completed i.e. monitoring is done off-line. Potential faults in the process can be detected from the process data and timely predictions can be made for product quality variables compared to physical analysis of these variables. Unfortunately, any corrections that need to be applied to the process will only affect subsequent batches. In industrial production, where multiple batches are run in rapid succession, the corrections may not even be able to be applied to batches that were already started before the data have been processed. For example, if a fault is detected in the data after application of the models and is diagnosed as contamination of one the reactants, then all batches that have been charged with the contaminated reactant are not likely to proceed along the satisfactory batch trajectories. Instantaneous monitoring of the data would be preferred to minimise the effects of these faults.

Process data analysis that can provide near real-time feedback to the operator must be done while the batch run is ongoing, or on-line. On-line monitoring provides almost instantaneous feedback on the process and if corrections to the current batch are able to be made, they can be. If faults are identified in the process and it is determined that the batch will not be able to produce a product of satisfactory quality, the batch can be discarded before it has run to completion. Monitoring in an on-line fashion may save time, which is valuable in industrial batch production.

1.3 Problem Identification and Study Objectives

Methods used for off-line monitoring are insufficient for use in on-line monitoring due to the nature of the incoming data and the speed at which the data must be processed for monitoring applications. The batch trajectory is not complete at the time when models need to be applied, therefore these techniques cannot use features of a complete batch trajectory. Monitoring in near-real time also requires timely processing of the data. Data have to be processed every time a new set of measurements is available; as a result, the computational expense of processing the data is a critical consideration while still expecting to ensure accurate results.

The aim of this work is to implement an on-line fault detection and end-of-batch quality prediction platform using latent methods, such as Principal Component Analysis (PCA) and Partial Least Squares (PLS). These methods are able to construct bilinear models to approximate the data, which are useful when the batch process variables have a linear correlation structure. However, these models could become inaccurate when the correlation structure between the variables changes, therefore it is of value to attempt to partition the data (into phases) based on linear correlation structure and apply separate models to each phase.

Preparation of the data via pre-processing is a necessary step before the models can be applied. Synchronisation of uneven batch trajectories is usually required and can be a computationally expensive endeavour. Furthermore, the identification of the which model should be applied to the data (if the data are partitioned) is necessary to produce meaningful results.

The objectives of this study are:

- To determine whether on-line synchronisation is able to produce computationally efficient results comparable to results obtained using off-line synchronisation
- To determine whether batch data could automatically be partitioned based on correlation structure
- To determine whether the on-line platform could provide accurate but timely monitoring and end-of-batch quality prediction results
- To compare the on-line fault detection and end-of-batch quality prediction performance of models trained on batch data partitioned based on correlation structure to conventional modelling procedures

1.4 Thesis Overview

The relevant theory pertaining to batch process monitoring and current state-of-the-art literature is reviewed in the next chapter. After review of the literature, the approach used in this thesis is described. These approaches are applied to three case studies in the following three chapters: A Three Tank Simulation (chapter 4), the Penicillin Cultivation Process (chapter 5) and the Final Concentrate Dissolution Process (chapter 6). Conclusions are drawn from the results the objectives are reviewed in chapter 7, and recommendations are made in chapter 8.

CHAPTER 2 Batch Process Monitoring Theory

2.1 Data Collection in Batch Processes

As per the definition of batch processes, the feed is charged into the vessel and removed sometime later. The recipe consists of known quantities of reagents that are added to the vessel along with certain initial operating conditions, which are chosen in order to obtain a product within a specific quality range. During the batch run, process variables are measured and recorded to gather information about how the process is evolving. At the end of the batch run, the quality of the product must be assessed so that it falls within a specified quality range. Some variables are quick and easy to measure (e.g. temperature can be measured easily at certain places using a temperature probe, which gives almost instant feedback). Other variables may be difficult to measure directly or may take time to be analysed, and thus feedback is not available for some time after the measurement was taken (e.g. the composition of material may need to be analysed in a laboratory, which could take time).

When collecting data during a batch run, a number of J measured variables are recorded. These variables can be arranged into a vector $\boldsymbol{x}$, with each scalar x:

$$\boldsymbol{x} = \left(x_1, x_2, \dots, x_j, \dots, x_J\right), \qquad \boldsymbol{x} \in \mathbb{R}^{1\times J} \tag{2.1}$$

The J variables are recorded for a number of K sampling times. This is done for a number of I batches that need to be processed. The batches may not have the same duration, so each batch may have a different number of sampling times. Synchronisation is done to align the trajectories to have the same number of sampling times.

The data can be arranged in a three-way matrix $\underline{\boldsymbol{X}} \in \mathbb{R}^{I\times J\times K}$ (figure 2.1). In the situation where different batches have a different number of sampling times, the data must be processed in some way to ensure that the number of sampling times is similar.

The scalar x_{ijk} refers to the j-th measured variable at a specific time k during a specific batch i where $i, j, k \in \mathbb{N}$.

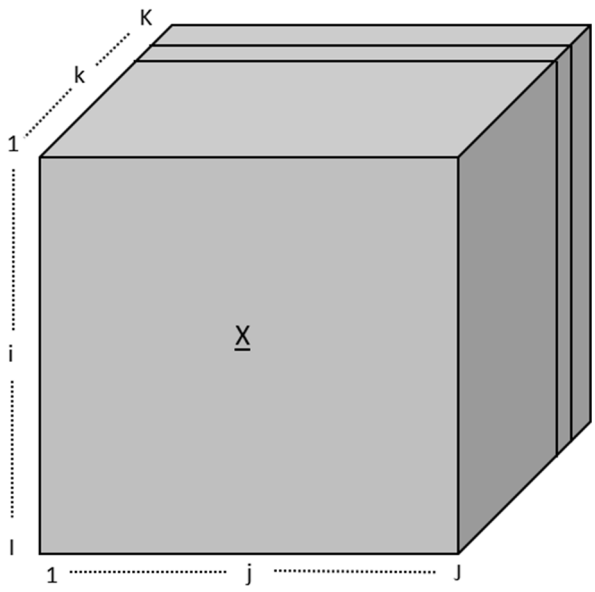


Figure 2.1: Three-way matrix $\underline{X}$

Data may also be collected for the variables that describe the quality of the product and similarly can be arranged in the matrix $\underline{\boldsymbol{Y}} \in \mathbb{R}^{I \times M \times K_y}$ for I batches, M quality variables and K_y sampling times. Usually these variables are only sampled at the end of the batch and the time mode can be dropped, resulting in a two-way matrix $\boldsymbol{Y} \in \mathbb{R}^{I \times M}$. Any other information that is important to the process is arranged in a matrix $\boldsymbol{Z} \in \mathbb{R}^{I \times R}$.

Not all batch runs will result in the desired product quality. It is important when building an historical database that only the batches that resulted in the desired product quality are used to train a reference model. These data are named normal operating conditions (NOC) data. This is because a data-based model uses the data to create a representation of the desired process outcome, and any data from batches that do not end in the desired quality will not form a sufficient representation of the desired process outcome.

2.2 Multivariate Statistical Modelling of Batch Processes

Batch data are three-way as they have three modes: the measured variables, the time when the measurement was taken and the batch that the measurement was taken from. In order to model three-way data, two approaches can be followed:

1) Applying two-way (or bilinear) models to the data. In order to do this, the three-way data must be rearranged into a two-way matrix, called unfolding. Nomikos and MacGregor (1994) successfully reported on Multiway Principal Component Analysis (MPCA) and extended their work by using Multiway Partial Least Squares (MPLS) to predict final product qualities (Nomikos & MacGregor, 1995).

2) Applying three-way (or trilinear) models to the data. These techniques apply models directly to three-way data. These models include Parallel Factor Analysis (PARAFAC), Tucker3 and Tucker2 models (Kroonenberg, 1992).

Three-way models have the advantage that the data matrix does not have to be rearranged before models can be trained. Bro (1996) mentions that these models are thus easier to interpret and potentially less prone to noise as the information for all modes is used in the decomposition of the matrix. Bro (1996) applied both three-way Partial Least Squares (tri-PLS) and two-way PLS models to fluorescence excitation-emission matrices in order to predict the ash content of sugar and found that the predictions from the tri-PLS model were more accurate than the two-way PLS model. However, only one unfolding technique (variable-wise unfolding – discussed in section 2.3.3) was used for the two-way model, which has been outperformed by other unfolding techniques (Camacho, et al., 2007).

Modelling two-way data, on the other hand, is much easier than modelling three-way data. Scaling and centring (section 2.3.4) the data is more complex for three-way models. Each mode has to be centred one at a time in a sequential manner to preserve the multilinear nature of the data (Bro, 1997). Scaling must also be performed across the entire mode (slab scaling) as opposed to each column (auto scaling) and scaling in more than one mode becomes quite complicated.

Recent studies have compared the different modelling approaches. Moita et al. (2014) evaluated the performance of several two- and three-way multivariate statistical methods in on-line batch processes and found that (two-way) Principal Component Analysis showed the best overall performance. Rato et al. (2016) proposed a comparison and assessment methodology to compare the fault detection strength of two- and three-way models. This methodology was employed to comprehensive data from two case studies: a penicillin cultivation process simulation and a semi-batch reactor simulation of an exothermic reaction. Adequate solutions were found with two-way models, however, three-way models were more sensitive to certain faults in the system. In general, no outright superior performance was observed for any modelling method.

The choice of which modelling approach is generally reliant on the data itself, which is why the comparison and assessment methodology was proposed by Rato et al. (2016). In this thesis, two-way models were used because of the lesser complexity of scaling and centring. In the following sections, the concepts of Principal Component Analysis (PCA) and Partial Least Squares (PLS) are introduced, thereafter their use in monitoring of batch processes is discussed.

2.2.1 Principal Component Analysis (PCA)

PCA seeks to reduce the dimensionality of a set of correlated variables into a smaller set of uncorrelated variables (Dunteman, 1989) and a residual term. Process data that are arranged in the two-way matrix $\boldsymbol{X} \in \mathbb{R}^{I \times J}$, where I is the number of objects and J is the number of observations, usually consists of many correlated observations (Kourti, 2002). The correlations between the observations indicate that most of the variance shown reflects the same disturbance or independent variable change. It would be useful to transform these variables into a set of uncorrelated variables, such that the variation amongst the data can be easier understood. Furthermore, because the original data are correlated, most of the uncorrelated data can be explained by much fewer variables.

PCA is affected by the magnitudes of the variables, hence all variables should be normalised in some meaningful way to negate this effect. For purposes of this discussion on the theory of PCA, it is assumed that the data matrix $\boldsymbol{X}$ has been normalised appropriately. Normalisation by centring and scaling is discussed in section 2.3.4.

PCA follows the general equation:

$$\boldsymbol{X} = \boldsymbol{T} \cdot \boldsymbol{P^T} + \boldsymbol{E} \tag{2.2}$$

Here, the loading matrix $\boldsymbol{P} \in \mathbb{R}^{J \times A}$ is a matrix that shows how the latent variables are related to the original data and the score matrix $\boldsymbol{T} \in \mathbb{R}^{I \times A}$ is a matrix of the values corresponding to these relations. The dimension A refers to the number of principal components chosen. $\boldsymbol{E} \in \mathbb{R}^{I \times J}$ is a matrix of the residuals that contains elements of the data that are not explained by the principal components. The values in $\boldsymbol{P}$ are found by finding the eigenvectors of the square matrix

$$\boldsymbol{X}^* = \frac{\boldsymbol{X^T} \cdot \boldsymbol{X}}{J-1} \in \mathbb{R}^{J \times J} \tag{2.3}$$

$\boldsymbol{X}^*$ is an estimation of the covariance matrix of $\boldsymbol{X}$ for mean centred data (Kourti, 2002). The eigenvectors represent the orthogonal directions of the principal components. If the principal components are orthogonal to one another, they are uncorrelated. Each eigenvector is arranged as a column in $\boldsymbol{P}$ and are found first by solving the equation

$$\det(\boldsymbol{X}^* - \lambda \cdot \boldsymbol{I}) = 0 \tag{2.4}$$

to get the J number of eigenvalues (λ). $\boldsymbol{I} \in \mathbb{R}^{J \times J}$ is the identity matrix. The eigenvectors can then be found by solving for $\boldsymbol{p_j}$ for each eigenvalue λ_j, resulting in J eigenvectors.

$$\left(\boldsymbol{X}^* - \lambda_j \cdot \boldsymbol{I}\right) \cdot \boldsymbol{p_j} = \boldsymbol{0} \tag{2.5}$$

The eigenvalues λ_j correspond to the amount of variance explained by its eigenvector $\boldsymbol{p_j} \in \mathbb{R}^{J \times 1}$. This is used to order the matrix $\boldsymbol{P}$. Only A eigenvectors are retained that explain most of the variance of $\boldsymbol{X}$. The choice of how many eigenvectors to retain can be done using cross validation (explained further in section 3.2.1) if the purpose of the model is to be used on future observations (Nomikos & Macgregor, 1995). The score vectors of each component are found using the equation

$$\boldsymbol{t_j} = \boldsymbol{X} \cdot \boldsymbol{p_j} \tag{2.6}$$

where $\boldsymbol{t_j} \in \mathbb{R}^{I \times 1}$. The A retained score vectors are arranged as columns in the matrix $\boldsymbol{T}$ from equation 2.2, resulting in a matrix with I rows and A columns. Equation 2.2 can also be written as

$$\boldsymbol{X} = \sum_{a=1}^{A} \boldsymbol{t_a} \cdot \boldsymbol{p_a}^T + \boldsymbol{E} \tag{2.7}$$

The Non-Linear Iterative Partials Least Squares (NIPALS) algorithm (Wold, 1974) can be used to calculate the first few principal components in order. The algorithm has value when the A number of principal components chosen is significantly less than the number of J observations, as the discarded principal components need not be calculated.

A new observation vector $\boldsymbol{x_{new}}$ can be projected onto the latent subspace given by the principal components, resulting in the new score vector $\boldsymbol{t_{new}}$:

$$\boldsymbol{t_{new}} = \boldsymbol{x_{new}}\, \boldsymbol{P} \tag{2.8}$$

where $\boldsymbol{t_{new}} \in \mathbb{R}^{1 \times A}$ and $\boldsymbol{x}_{\boldsymbol{new}} \in \mathbb{R}^{1 \times J}$.

2.2.2 Partial Least Squares (PLS)

The goal of partial least squares is to find a linear regression model between two sets of data by using the relations between the two data sets. The data of the second set can then be predicted by the data in the first set. This is done by projecting both data sets onto the same latent subspace and subsequently finding the regression matrix. Ordinary least squares (OLS) does not find such a latent subspace but rather finds a regression model directly between the two data sets. PLS is

preferable to OLS when the variables in the first set are highly correlated (Camacho, et al., 2007), as is often the case in with batch process variables.

In process analysis, PLS can extract the latent variables that explain the variation in the process data that are most predictive of the quality data (Kourti, 2002). Linear models of the normalised process and quality matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ can be written

$$\boldsymbol{X} = \boldsymbol{T}\boldsymbol{P}^{\boldsymbol{T}} + \boldsymbol{E} \tag{2.9}$$

where $\boldsymbol{X} \in \mathbb{R}^{I \times J}$, $\boldsymbol{T} \in \mathbb{R}^{I \times A}$ $\boldsymbol{P} \in \mathbb{R}^{J \times A}$, $\boldsymbol{E} \in \mathbb{R}^{I \times J}$ and

$$\boldsymbol{Y} = \boldsymbol{T}\boldsymbol{Q}^{\boldsymbol{T}} + \boldsymbol{F} \tag{2.10}$$

where $\boldsymbol{Y} \in \mathbb{R}^{I \times M}$, $\boldsymbol{Q} \in \mathbb{R}^{M \times A}$, $\boldsymbol{F} \in \mathbb{R}^{I \times M}$. The matrices are defined for a number of I objects, with J observations and M quality variables. The scores matrix $\boldsymbol{T}$ contains the projections of $\boldsymbol{X}$ to the latent subspace, respectively. $\boldsymbol{P}$ and $\boldsymbol{Q}$ are the loading matrices that show how the latent variables are related to the original matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively. The latent subspace has A mutually independent dimensions. $\boldsymbol{E}$ and $\boldsymbol{F}$ are the residual matrices of $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively. As for PCA, it is assumed that the data in $\boldsymbol{X}$ and $\boldsymbol{Y}$ are scaled appropriately.

In the NIPALS algorithm, the first weight vector $\boldsymbol{w_1}$ is the first eigenvector of the sample covariance matrix (Höskuldsson, 1988)

$$(\boldsymbol{XY})^* = \boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{Y}\boldsymbol{Y}^{\boldsymbol{T}}\boldsymbol{X}, \qquad (\boldsymbol{XY})^* \in \mathbb{R}^{J \times J} \tag{2.11}$$

The scores for this component can then be found by

$$\boldsymbol{t_1} = \boldsymbol{X}\boldsymbol{w_1} \in \mathbb{R}^{I \times 1} \tag{2.12}$$

The first loading vector $\boldsymbol{p_1}$ of the matrix $\boldsymbol{P}$ can then be calculated from the scores by using the method of least squares which gives

$$\boldsymbol{p_1} = \frac{\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{t_1}}{\boldsymbol{t_1}^{\boldsymbol{T}}\boldsymbol{t_1}} \in \mathbb{R}^{J \times 1} \tag{2.13}$$

These vectors form a linear model of the $\boldsymbol{X}$ matrix

$$\boldsymbol{X} = \boldsymbol{t_1}\boldsymbol{p_1}^{\boldsymbol{T}} + \boldsymbol{X_2} \tag{2.14}$$

The residual matrix $\boldsymbol{X_2}$ contains all the values of $\boldsymbol{X}$ not explained by the first prediction. Using the NIPALS method, these residuals are used to get a second prediction. Similarly, the first loading vector $\boldsymbol{q_1}$ of the matrix $\boldsymbol{Q}$ can be found using the method of least squares

$$\boldsymbol{q_1} = \frac{\boldsymbol{Y^T t_1}}{\boldsymbol{t_1}^T \boldsymbol{t_1}} \in \mathbb{R}^{M\times 1} \tag{2.15}$$

The linear model of the $\boldsymbol{Y}$ matrix then becomes

$$\boldsymbol{Y} = \boldsymbol{t_1 q_1}^T + \boldsymbol{Y_2} \tag{2.16}$$

The second weight vector $\boldsymbol{w_2}$ is the first eigenvector of the new covariance matrix of the residual matrices $\boldsymbol{X_2}$ and $\boldsymbol{Y_2}$:

$$\boldsymbol{X_2}^T \boldsymbol{Y_2}\, \boldsymbol{Y_2}^T \boldsymbol{X_2} \in \mathbb{R}^{J\times J} \tag{2.17}$$

The scores $\boldsymbol{t_2}$ and loading vector $\boldsymbol{p_2}$ for this component are similarly calculated. They can be used to explain some of the variance not explained by the first prediction. This process is repeated for the number of A mutually independent dimensions. A disadvantage to this approach is that each weight vector applies to a deflated $\boldsymbol{X_a}$ matrix, therefore it is difficult to calculate the scores from the weight matrix. An alternative approach to calculating weight matrix ($\boldsymbol{R}$) that is directly applicable to $\boldsymbol{X}$ is to use the SIMPLS algorithm (de Jong, 1993). Using this algorithm, the scores matrix can be expressed as

$$\boldsymbol{T} = \boldsymbol{XR} \tag{2.18}$$

In general, the PLS model can be written as:

$$\boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{F} \tag{2.19}$$

where $\boldsymbol{\beta} \in \mathbb{R}^{J\times M}$ is the regression coefficient matrix and is given by

$$\boldsymbol{\beta} = \boldsymbol{RQ}^T \tag{2.20}$$

and $\boldsymbol{R} \in \mathbb{R}^{J\times A}$ is a matrix of the weight vectors that relate the scores to the input matrix.

2.2.3 Process Monitoring and End-of-Batch Quality Prediction

The methods above can be extended for use in process monitoring. Models are built using the methods above to relate the process data matrix $\boldsymbol{X}$ and/or the quality data matrix $\boldsymbol{Y}$ under normal operating conditions. Two statistics are used when monitoring processes using latent methods (Kourti & MacGregor, 1996).: The Hotellings T^2 statistic on the first A latent variables for a sample

$$T_A{}^2 = \sum_{a=1}^{A} \frac{t_a{}^2}{s_{t_a}} \tag{2.21}$$

where s_{t_a} is the estimated variance of the corresponding latent variable and t_a is the score of the a-th component of the sample. This statistic measures the distance from the origin of the hyperplane defined by the first A components. The second statistic used is the SPE_x statistic, which calculates the sum squared error between the actual values of $x_{new,j}$ and its reconstructed value, $\hat{x}_{new,j}$ across all J samples:

$$SPE_x = \sum_{j=1}^{J} \left(x_{new,j} - \hat{x}_{new,j}\right)^2 \tag{2.22}$$

The control limits for $T_A{}^2$ can be calculated as the critical value of the F-statistic of A and I-A degrees of freedom at significance level α times a constant (Camacho, et al., 2007):

$$(T_A{}^2)_\alpha = \frac{A(I^2-1)}{I(I-A)} F_{(A,I-A)_\alpha} \tag{2.23}$$

The control limit for the SPE_x statistic can be calculated using the approximation for the chi-squared variable at significance level α (Nomikos & Macgregor, 1995):

$$SPE_\alpha = gh\left[1 - \frac{2}{9h} + z_\alpha \left(\frac{2}{9h}\right)^{1/2}\right]^3 \tag{2.24}$$

where

$$g = \frac{s_{SPE}}{2\mu_{SPE}} \tag{2.25}$$

and

$$h = \frac{2\,(\mu_{SPE})^2}{s_{SPE}} \tag{2.26}$$

Here, μ and s are the sample mean and variance of the SPE sample at each time point, and z_α is the normal variable at significance level α. If the T_A^2 or SPE_x exceed these defined limits, an alarm is recorded. A true alarm occurs when one of the control limits is exceeded and a fault has occurred in the process. A false alarm occurs when a control limit is exceeded and no fault occurred in the process. A missed alarm occurs when a fault has occurred in the process but no control limit was exceeded. The detection delay can also be defined as the time between a fault occurring and three consecutive true alarms.

Quality can be predicted by constructing a PLS model of the process under normal operating conditions. The quality variables $\hat{\boldsymbol{y}}_{\boldsymbol{new}} \in \mathbb{R}^{1\times M}$ may then be predicted using the incoming process data $\boldsymbol{x}_{\boldsymbol{new}} \in \mathbb{R}^{1\times J}$ by

$$\hat{\boldsymbol{y}}_{\boldsymbol{new}} = \boldsymbol{x}_{\boldsymbol{new}}\,\boldsymbol{\beta} \qquad \textbf{2.27}$$

The confidence levels of a prediction of $\hat{\boldsymbol{y}}$ have been suggested by Nomikos and MacGregor (1995):

$$\hat{\boldsymbol{y}} \pm t_{I-A-1,\alpha/2} \cdot (MSE)^{1/2} \cdot (\mathbf{1} + \hat{\boldsymbol{t}}\,(\boldsymbol{T}^{\boldsymbol{T}}\,\boldsymbol{T})^{-\mathbf{1}}\,\hat{\boldsymbol{t}}^{\,\boldsymbol{T}})^{1/2} \qquad \textbf{2.28}$$

where $t_{I-A-1,\alpha/2}$ is the critical value of the studentised t-statistic with *I-A-1* degrees of freedom based on the model generated from *I* training batches and *A* retained latent variables at confidence level $\alpha/2$ and MSE is the Mean Squared Error of the model. $\boldsymbol{T}$ is the scores matrix of the training data and $\hat{\boldsymbol{t}}$ is the estimated scores vector of the incoming batch calculated by

$$\hat{\boldsymbol{t}} = \boldsymbol{x}_{\boldsymbol{new}}\,\boldsymbol{R} \qquad \textbf{2.29}$$

The use of these two-way models to describe three-way data is appropriate only if the data are pre-processed effectively. Pre-processing is an important and sometimes computationally demanding exercise in batch process monitoring. The following section describes the pre-processing steps required for analysis of batch data in both off- and on-line monitoring.

2.3 Pre-processing of Batch Data

Multivariate statistical process control methods have been adapted to batch processes by means of a variety of data pre-processing techniques. Pre-processing models are first trained from NOC data and then applied to new data. First the concepts of phase identification, synchronisation and data unfolding are introduced; thereafter the procedures for pre-processing of training models and applying these pre-processing models to new data are presented.

2.3.1 Phase Division and Identification

Batch processes may go through distinct phases as they evolve with time. The covariance structure of the variables may vary between these phases, thus one model may not be sufficient to model the whole process. Ündey & Çinar (2002) define a multiphase process as a batch process with a single processing unit but multiple operating regimes. This is distinct from a multistage process, which involves having multiple processing units. Camacho, et al. (2008) defines a phase as a segment of the batch duration that is well approximated by a single linear model. Although these definitions are not absolutely consistent, they are also not mutually exclusive. The former definition describes the physical process, whereas the latter describes the modelling of the data given by the process. Henceforth, to distinguish between these definitions, the term "correlation phases" will be used when referring to phases described by the Camacho, et al. (2008) definition. When referring to phases described by the Ündey & Çinar (2002) definition, the term "operating phases" will be used. The term "phase" (and subsequently "phase division" and "phase identification") encompasses both these definitions.

Monitoring results obtained when applying the models to new data are only as accurate as the models that were trained. Fitting just one model for the entire batch duration may show reasonable overall performance, but provide poor predictions during certain shorter phases of the process. The poor performance of the model during shorter phases is masked by the performance during the longer phases as there are more observations in the longer phases. Faults occurring during these poorly modelled phases won't be detected until the model is able to explain the data better. It becomes necessary to divide the process into separate phases when it can be assumed that a statistical model applied to each individual phase will yield better performance than the same model applied to the entire batch trajectory.

When modelling data, the relevant phases first need to be identified and the data partitioned accordingly. Subsequently, models can be trained for each phase. In order to apply the correct model to the new data it is first necessary to identify what phase the data being analysed corresponds to. Once the working phase has been identified, the data can be partitioned and the

model corresponding to the identified phase can be applied. A variety of techniques on how to divide and identify phases have been discussed (Yao & Gao, 2009) and will be reviewed in detail in section 2.4.

2.3.2 Synchronisation

In general product quality, rather than processing time, is the deciding factor for the successful completion of a batch. As a result, the batch durations may differ and the batch trajectories are not always equal. Before multivariate statistical modelling methods for batch process monitoring and quality prediction can be applied, the data must be aligned through synchronisation. Kourti (2006) states that synchronisation of the data must achieve the following:

1) Establish common start points for different phases of the run
2) Match the shape of the trajectories of the variables

A statistical model describes a relationship between variables. It is important that the model describes the relationship between the variables at the same part of the process for each batch. The shapes of the trajectories represent the progress of the batch, so it follows that the shape of the trajectories should be matched via synchronisation while still retaining batch-specific information. The necessity to establish common start points for the different phases follows from the need to apply different models to different phases, as discussed in the previous section. Once synchronisation has been completed, the batches will share a common pseudo-time (which may correspond to a reference batch time, although this is not a pre-requisite). Additionally, due to the distortions, the values of some synchronised data are not the actual measurements of the process data (Kourti, 2002). These techniques must be applied carefully such that the auto- and cross-correlations of the data are not seriously affected.

Figure 2.2 shows the trajectories of two batches, indicated by the dotted and solid lines. In a), the trajectories are similar with one ending earlier than the other. Simple clipping of the excess data in the longer trajectory is appropriate to synchronise these trajectories. In b), the trajectories are similar, but are shifted in time by a constant factor. Simple shifting of these trajectories in time will synchronise these trajectories. However, in c) and d), the simple techniques used for a) and b) cannot be used and more complex techniques need to be used to align the data. Such techniques for synchronisation of batch trajectories are reviewed in section 2.4.

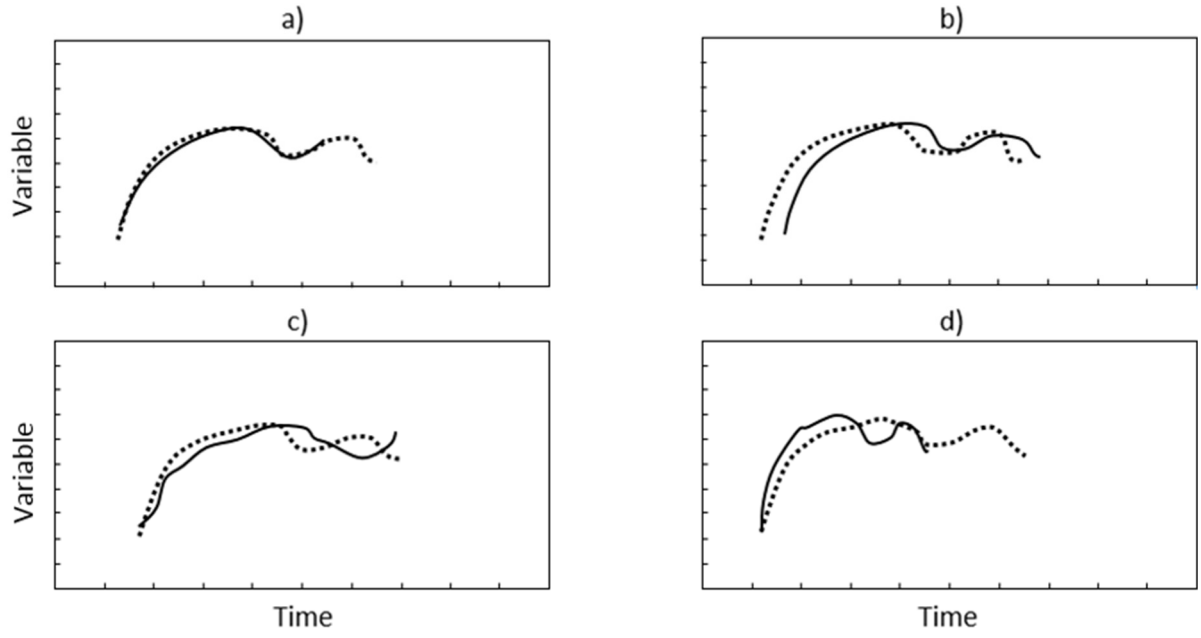


Figure 2.2: Comparison of two batch trajectories. Redrawn from Kourti (2002)

2.3.3 Data Unfolding

Bilinear models can only be applied to data arranged in a two-way matrix. Batch process data are three-way, having axes for the batches, process variables and time. Unfolding of the data seeks to rearrange the three dimensional data in such a way that two-way models can be applied.

Two traditional methods have been used to unfold the data. Nomikos and MacGregor (1994) used the batch-wise unfolding method to rearrange the batch data. This method involves putting each of the vertical slices of the three-way matrix side by side to create the matrix $\boldsymbol{X_{BW}} \in \mathbb{R}^{I \times JK}$. This is shown in figure 2.3 a). Wold et al. (1998) used the variable-wise unfolding method by putting each horizontal slice of the matrix one under another to create the matrix $\boldsymbol{X_{VW}} \in \mathbb{R}^{IK \times J}$ (figure 2.3 b).

The resulting covariance matrices (equation 2.3 for PCA and equation 2.11 for PLS) of each of these methods of unfolding reveal differences in the information contained by a model. The covariance matrix for batch-wise unfolded data is of size $\boldsymbol{X^*_{BW}} \in \mathbb{R}^{JK \times JK}$. This is because each variable at each individual sampling point is treated as a separate observation. Therefore, it not only takes into account the variances and instantaneous cross-covariances but also the auto-covariances and lagged cross-covariances of the variables (i.e. relationship between variables at different times in the process). Using this information, Camacho et al. (2007) concluded that batch-wise unfolding does incorporate the linear dynamics of the process.

The covariance matrix for variable-wise unfolded data is of size $\boldsymbol{X^*_{VW}} \in \mathbb{R}^{J \times J}$. This covariance matrix can be represented by a sum of the individual covariance matrices (only variances and instantaneous cross-correlations) for each time, divided by the factor K. This is due to the fact that the variables over the entire sampling time are looked at as observations. This form of unfolding does not incorporate the auto-covariances and lagged cross-covariances. Additionally, because the covariance matrix is represented by a sum of the individual covariance matrices divided by a factor K, the relationship between a pair of variables in the covariance matrix is an average of their relationship throughout the batch duration. A second side-effect of this phenomenon is that missing variable measurements at certain time intervals for all batches during the process (often the case for real-world data) cannot be handled when generating one model of variable-wise unfolded data as the correlation structure of the process has changed (Kourti, 2003a).

Some authors have proposed additional unfolding methods. Ramaker et al. (2005) discussed the building of a separate local model for each of the time steps, which does not require unfolding as the data at each time step are two-way ($\boldsymbol{X_k} \in \mathbb{R}^{I \times J}$). If the data from previous local models are incorporated in the new model, it is called an evolving model. This may be a computationally expensive exercise as a separate model has to be built for every new sample point k. The method can be fine-tuned by selecting the number of components for each time interval, but this may not always be straightforward. Chen & Liu (2003) discuss batch dynamic unfolding, which recruits lagged measurement vectors (LMVs). These LMVs help incorporate both the static and dynamic process characteristics when creating models. A thorough analysis on the different unfolding techniques is given by Camacho et al. (2007), who found that different unfolding methods may be suitable for modelling different processes or parts thereof.

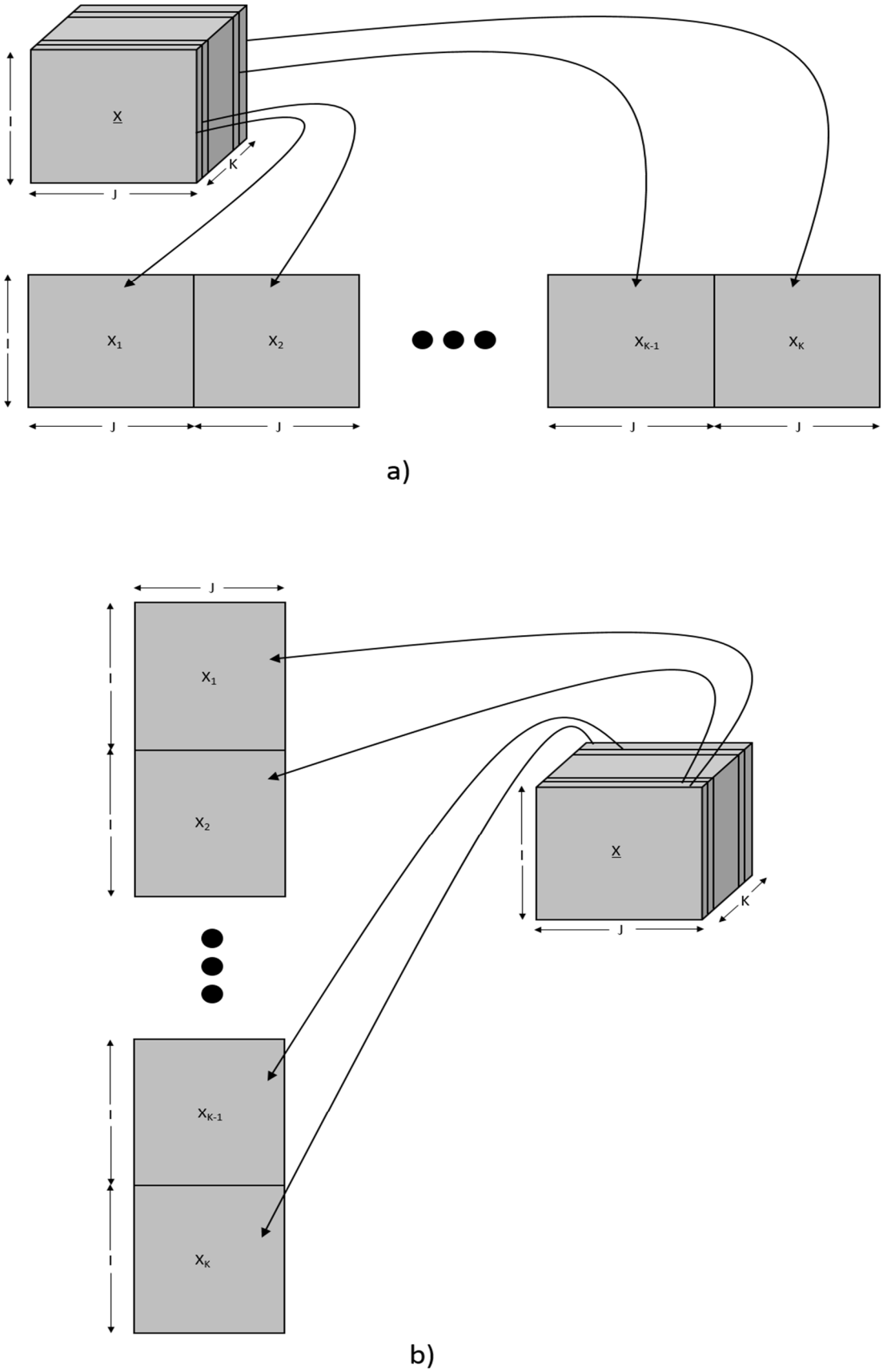


Figure 2.3: a) Batch-wise unfolding and b) Variable-wise unfolding. Redrawn from Camacho et al. (2007)

2.3.4 Centring and Scaling

One of the goals of statistical process monitoring is to check for deviations in the variable trajectory from the average trajectory (of the NOC training data). The actual trajectory itself is not a necessary part of the modelling. It would be appropriate to subtract the mean trajectory from the data before building a model, leaving behind only the deviations. This has the added advantage that it will reduce a non-linear trajectory to a linear one with a mean of zero. Such a technique is called centring (figure 2.4). Subtracting the mean in batch-wise unfolded data would subtract the mean of the variable at every time point, whereas subtracting the mean for variable-wise unfolded data would subtract the mean of the variable for the entire length of the batch run (i.e. the grand mean). Therefore, centring variable-wise unfolded data this way would not be able to remove non-linearities from the data.

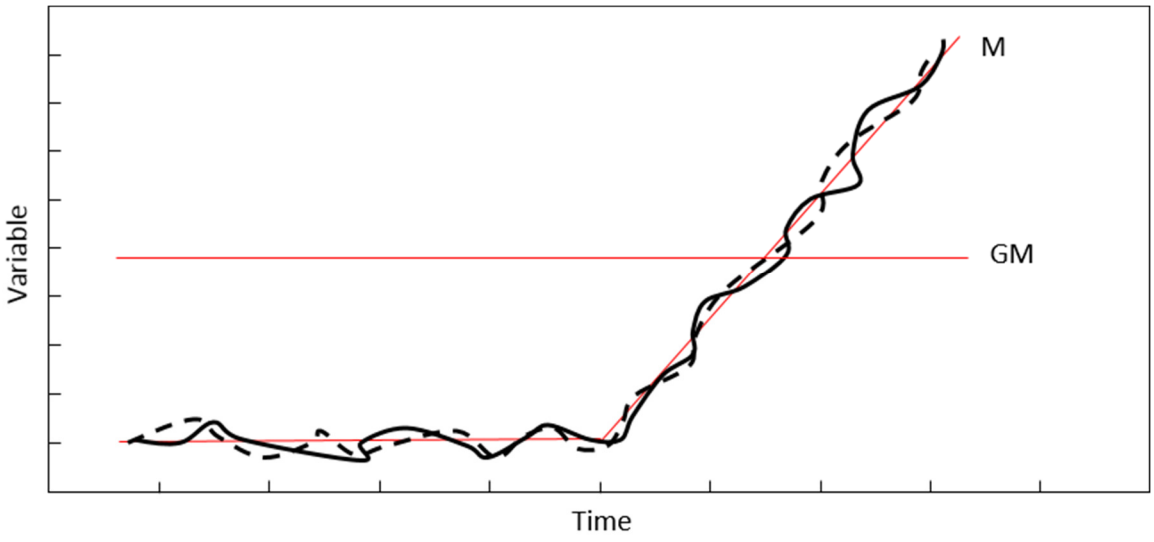


Figure 2.4: Two batch trajectories shown along with the mean (solid red line M) and grand mean (solid red line GM). The mean accounts for the non-linearities in the data, whereas the grand mean does not

Another problem that exists when monitoring different types of variables is that the magnitudes of these variables may differ. For example, the temperature (measured in °C) and the sugar concentration (measured in g/L) of a solution in a fermentation vessel have different scales. The deviations about the mean may not be of the same magnitude, therefore some of the variables may be given more weight than the others when modelling the process, primarily due to their magnitudes. The variables are given equal weight by scaling the variables. Kourti (2002) gives two methods of scaling data: scaling to unit variance by dividing each variable by its standard deviation at each point in time (auto scaling), or dividing by its standard deviation across the whole time (slab scaling). The former approach is preferred because in the latter approach, periods of high noise will be weighted more and periods under tight control will be weighted less. However, auto scaling can only be applied to synchronised data.

2.3.5 Incomplete Data Imputation

Incomplete data can be categorised in two ways:

1. ***Missing Data***. Data that were not recorded during the batch run for any of a number of reasons. At a certain time interval, if only certain measured variables are able to be recorded, the unrecorded data is treated as missing data.
2. ***Future Data***. Data that have yet to be recorded, as is the case in on-line monitoring when the batch run has not reached completion.

Methods of treating missing data for monitoring purposes were reviewed by Arteaga and Ferrer (2002), and superior performance was found with known data regression (KDR), where missing data are estimated using the training data; and trimmed score regression (TSR), where missing data are estimated using the trimmed scores i.e. the scores with zero deviations. Kourti (2003b) mentions that if there are missing data measurements for certain variables at certain time intervals throughout all batches of the process, these missing data do not have be included when training the corresponding model (the size of the unfolded matrix will change as a result). Kourti (2003a) further explains that treating these missing data using missing data procedures during modelling may not be correct practice as the missing values were not actually observed and as a result will affect the accuracy of the model.

Future data should also be addressed. After on-line synchronisation, data monitored would be aligned only up to its corresponding point on the reference trajectory. In order to use multivariate statistical methods, the dimensions of the input matrices must be preserved. In batch-wise unfolded data, the dimensions of the modelled data are $\boldsymbol{X_{BW}} \in \mathbb{R}^{I \times JK}$ i.e. each variable at each time is seen as a new observation. The synchronised, unfolded matrix would only have the dimensions $\boldsymbol{x_{new,BW}} \in \mathbb{R}^{1 \times Jk}$, where $k < K$. The matrix must be padded with values in order for the dimensions of the second mode to be similar to modelled data matrix. Alternatively, if the data are unfolded variable-wise ($\boldsymbol{X_{VW}} \in \mathbb{R}^{IK \times J}$), the incomplete data do not have to be added as it won't affect the size of the resulting covariance matrix. The methods for treating missing data can be used to pad the batch-wise unfolded matrix.

Synchronisation and phase identification for on-line use must also be considered with the above challenges in mind. The following sections review such synchronisation and phase identification techniques proposed in the literature with emphasis to these challenges.

2.4 Review of Synchronisation Techniques

The problem at hand regarding synchronisation is aligning trajectories in batches that do not proceed at the same rate. Figure 2.2 shows example cases when synchronisation is necessary. One of the goals of synchronisation as mentioned in section 2.3.2 is to match the shape of the trajectories of the variables without losing batch specific information. In off-line monitoring, this can be done by renormalizing the time scale so that all batches have the same duration (Nomikos & MacGregor, 1995). However, this is not feasible for on-line monitoring, because the whole duration of the batch is not known beforehand. Additionally, some parts of the process may proceed faster than other parts of the process over the course of a NOC batch run compared with another NOC run. A linear transformation of the time axis for the whole run is therefore not appropriate (Westerhuis, Kourti, & Macgregor, 1999). One can divide the process into stages or phases, but again this is only possible once the entire phase or stage has been completed. The approaches that follow have been used in order to synchronise unequal batch trajectories.

2.4.1 Indicator Variable (IV) Approach

The definition of unequal batch trajectories infers that time is not well correlated with the batch trajectory progress. Instead of using time as an indication for the progress of the process, Nomikos and MacGregor (1995) used a different variable. The criteria that such a variable (henceforth referred to as an indicator variable or IV) must satisfy are:

- The indicator variable must progress monotonically with time. This criterion ensures that the progress of the batch can be tracked as a function of the IV without duplicate measurements.
- The indicator variable must have the same starting and end point for each batch. These can be interpreted as global constraints on the system.

Such a variable is usually specific to the process, therefore some process knowledge is needed. Examples of an indicator variable include the on-line measure of conversion in a chemical reactor or the lance position in injection molding. Kourti et al. (1996) used the percentage amount of a certain ingredient fed to the reactor as the indicator variable in the industrial application of projection methods (PCA and PLS) to a semi-batch polymerisation process. One interval corresponded to 1% of the mass of the ingredient fed to the reactor. This variable satisfied both criteria as:

- The percentage of ingredient fed to the reactor increased with time (as nothing was removed until the semi-batch process was finished).

- The total amount of the ingredient fed to the reactor was kept constant. The starting point (nothing fed) and the endpoint (total amount fed) were the same for each batch.

This method is easy to implement, as all that is needed is to replace the time axis in the three-way matrix with the axis corresponding to the indicator variable (i.e. *I* number of batches with *J* number of process variable measurements over *K* number of indicator variable intervals). However, a variable that satisfies both the conditions presented above may not always exist (over the whole process or at certain stages or phases), or it may not be obvious which variable will be the best. Kassidas et al. (1998) mentioned that the weighting matrix used in the dynamic time warping algorithm can be used to find the most consistent variable, which may be a suitable indicator variable (which is discussed in the following section).

2.4.2 Dynamic Time Warping (DTW) Approach

In an attempt to overcome the shortcomings of the indicator variable approach, Kassidas et al. (1998) proposed using dynamic time warping to synchronise batch trajectories. Dynamic time warping is a method that is able to translate, compress and expand patterns locally and in a non-linear fashion so that similar features within patterns are matched. The theory of the original approach is documented below. Following this, the limitations of the method are discussed and the modifications to the original DTW approach are reviewed. A conceptual representation of DTW is shown in figure 2.5.

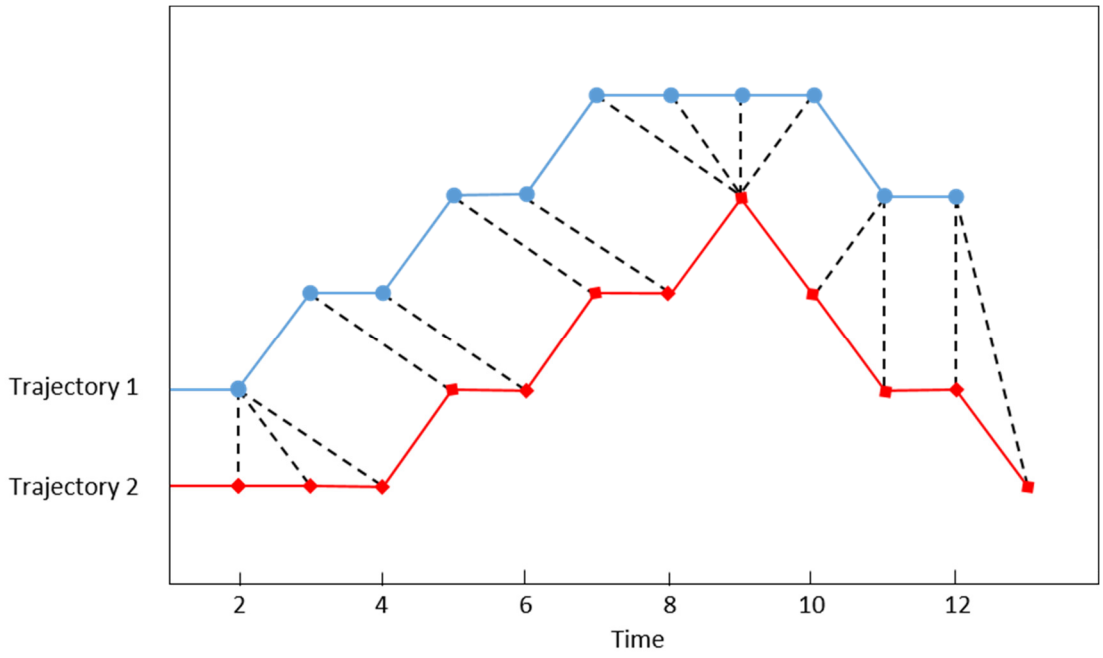


Figure 2.5: Conceptual representation of symmetric time warping. Matching points of two different trajectories are synchronised with one another along the dashed lines

Let $\boldsymbol{X_n} \in \mathbb{R}^{K_n \times J}$ be a set of trajectories of J variables of scaled, good quality batches with K_n time intervals for the i-th batch ($i \in \mathbb{N}^N$); and $\boldsymbol{X_{ref}} \in \mathbb{R}^{K_{ref} \times J}$ be a reference trajectory. Scaling can be done by a variety of methods, but Kassidas, et al. (1998) chose to divide by the average range of each variable in its trajectory. Scaling by average standard deviation or average interquartile range are also possible.

Let i and j denote the time index of the $\boldsymbol{X_n}$ and $\boldsymbol{X_{ref}}$ trajectories, respectively. DTW finds a sequence $\boldsymbol{F^*}$ of K_{warp} points on the $K_n \times K_{ref}$ grid (henceforth referred to as a synchronisation grid, shown in figure 2.6):

$$\boldsymbol{F}^* = \{c(1), c(2), \dots, c(k), \dots . c(K_{warp})\}, \qquad max(K_{ref}, K_n) \leq K_{warp} \leq K_{ref} + K_n \qquad \textbf{2.30}$$

where

$$c(k) = [i(k), j(k)], \qquad k \in \mathbb{N}^{K_{warp}} \qquad \textbf{2.31}$$

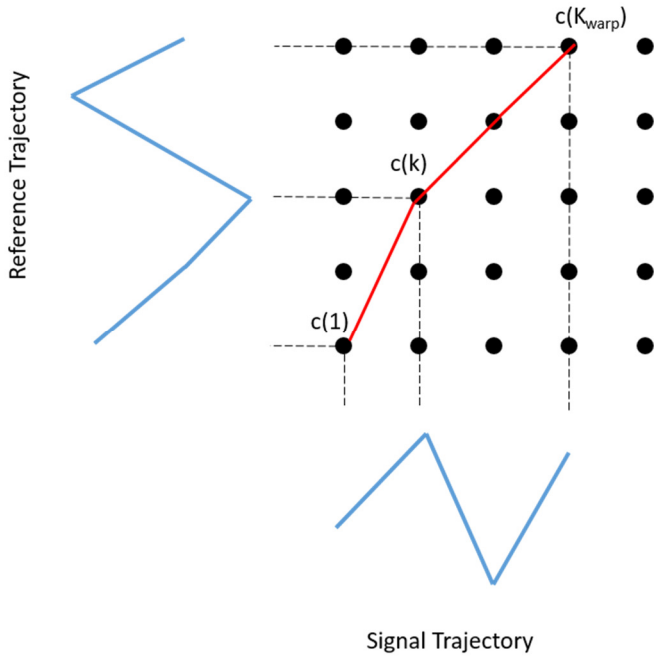


Figure 2.6: Synchronisation grid showing path taken when warping two trajectories

and each point $c(k)$ is a position in the grid indicating the elements of the two trajectories that are matched. DTW tries to find the minimum, total, normalised distance between the vectors that represent the two trajectories. Once the optimum path is found, the original trajectories will have been warped to correspond with one another while still retaining some batch-specific variation.

This can be done either symmetrically or asymmetrically. Symmetric DTW warps both trajectories to a new time axis, whereas asymmetric DTW preferentially warps one trajectory onto another. Asymmetric DTW makes more sense in terms of batch process monitoring as a reference batch is chosen and it would be sensible to warp all incoming trajectories onto this reference batch. However, Kassidas, et al. (1998) points out that some of the features of the original incoming trajectory that are inconsistent with the reference trajectory may be left out during asymmetrical warping, and this may bias it toward false alarms when an MPCA or MPLS model is applied.

Equation 2.30 finds numerous possible paths that are taken to advance along the grid, and DTW tries to find the optimum path. In order to do this and to narrow down the number of paths being considered, constraints should be given. This ensures that the algorithm searches for the optimal path in a more reasonable search space. Constraints that can be applied are:

1) ***Endpoint Constraints***. These constraints state that both trajectories should start and/or end at the same place on the grid. The initial endpoint is easy to determine as both trajectories would be constrained to start at $c(1) = [1,1]$. The final endpoint of both trajectories may not have to match each other exactly if there is an allowable region that the variable may end at. Additionally, the final condition can generally only be specified if the incoming trajectory data are known for the final condition. This is not the case in on-line monitoring.
2) ***Local Continuity Constraints***. The synchronisation grid maps the points of the two trajectories in time. It therefore follows that the desired path should follow a monotonic curve as time is monotonic. This will narrow the directions from with consecutive points can be reached. Additionally, a constraint that limits excessive compression and expansion can be imposed. This is done by specifying what previous points can be linked to the current point. The effect this has is that the gradient of the path must lie in a specific range. Itakura (1975) and Sakoe & Chiba (1978) define some local continuity constraints that can be applied in DTW for both symmetric and asymmetric algorithms (figure 2.7).
3) ***Global Constraints***. These constrain the whole set of points on the grid such that the path cannot deviate out of this range. The selection of local continuity constraints may also implicitly generate a set of global constraints; thus they may not always need to be specified. One of the most common global constraints is the Sakoe-Chiba band constraint, where the search space is reduced to exclude areas on the grid outside of a fixed-width band running parallel to the linear path. Ratanamahatana & Keogh (2004) proposed a learned band constraint that adjusted the width of the band at different parts of the process.

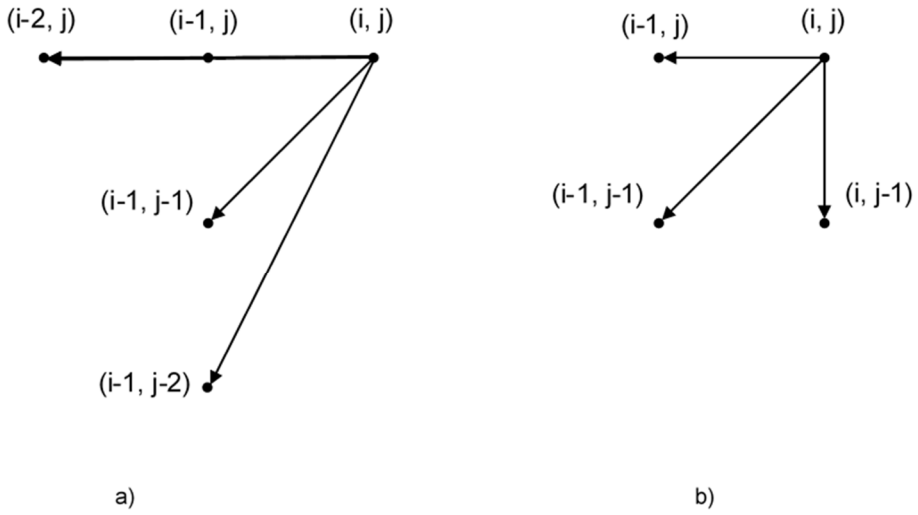


Figure 2.7: Itakura local constraint with slope [2, ½] and Sakoe-Chiba local constraint with no limitation on slope

Figure 2.7 shows two typical local constraints. The first is the Itakura constraint. It constrains the path such that the only way to reach the point (i,j) is through $(i-2,j)$, $(i-1,j-1)$ or $(i-1,j-2)$. Additionally, two consecutive horizontal transitions are not allowed. This results in a maximum gradient of 2 and a minimum gradient of ½ (denoted by the [2, ½]) in the figure. The second constraint is the Sakoe-Chiba constraint with no constraint on the slope. With this local constraint, global constraints may need to be applied. In this case, the deviation from the linear path starting at point [1,1] should be at least equal to the magnitude of the distance between the number of K_n and K_{ref} intervals.

In order to find the optimum path, the cumulated distances for each path need to be determined. The normalised cumulated distance between the two trajectories is

$$D(K_{ref}, K_n) = \frac{\sum_{k=1}^{K_{warp}} d(i(k), j(k)) \cdot w(k)}{N(w)} \qquad \textbf{2.32}$$

Here, $d(i(k), j(k))$ are the local distances between the variables at the k-th point on the grid (equation 2.32), $w(k)$ is a nonnegative normalization weight for the local distance and $N(w)$ is a normalisation factor corresponding to the normalisation weight. Kassidas et al. (1998) pointed out that for the application of DTW to synchronisation of batch trajectories, the normalisation factor plays no role and can be omitted because the number of reference observations is the same.

The local distance between each point on the grid is assessed by the weighted quadratic distance

$$d(i(k), j(k)) = \left[\boldsymbol{x}_{\boldsymbol{n,i(k)}} - \boldsymbol{x}_{\boldsymbol{ref,j(k)}}\right] \boldsymbol{W} \left[\boldsymbol{x}_{\boldsymbol{n,i(k)}} - \boldsymbol{x}_{\boldsymbol{ref,j(k)}}\right]^T \qquad \textbf{2.33}$$

where $\boldsymbol{x_{n,i(k)}} \in \mathbb{R}^{1\times J}$ and $\boldsymbol{x_{ref,j(k)}} \in \mathbb{R}^{1\times J}$ are the $i(k)$-th and $j(k)$-th row vectors of the $\boldsymbol{X_n}$ and $\boldsymbol{X_{ref}}$ trajectory matrices respectively. $\boldsymbol{W} \in \mathbb{R}^{J\times J}$ is a positive diagonal weight matrix that reflects the importance of each measured variable based on a particular criterion (this is distinct from the normalisation weight $N(w)$ in equation 2.32). Kassidas et al. (1998) proposed an iterative procedure to give more weight to those variables that are consistent from batch to batch. Ramaker et al. (2003) identified a disadvantage in this approach when dealing with horizontal lines in a trajectory as they contain no information that will assist warping, yet are constant and will dominate the weight matrix. The authors proposed a new weighting function that gave more importance to the variables with warping information (e.g. curved trajectories). González-Martínez et al. (2011) used the geometric mean of the two weighting functions so as to retain the features of both.

The minimum normalised distance is given by the solution to the optimisation problem, where F is the sequence of K_{warp} points on the grid:

$$\boldsymbol{D}^*(K_{ref}, K_n) = \min_F[D(K_{ref}, K_n)] \qquad \textbf{2.34}$$

Equation 2.34 can be solved, using dynamic programming, whereby the optimal predecessor of each point is stored in a vector, as well as the cumulated distance up until that point. Once the lowest cumulated distance is required, the optimal predecessor can be found as well, which links to the next optimal predecessor and so on and so forth, until the beginning of the grid is reached. In the dynamic time warping algorithm proposed by Kassidas, et al. (1998), symmetric DTW with fixed-endpoint constraints, band global constraint and Sakoe-Chiba local constraint (figure 2.7) is used first to warp both the $\boldsymbol{X_n}$ and $\boldsymbol{X_{ref}}$ trajectories to a common time axis. The optimisation problem for the symmetric DTW is:

$$\boldsymbol{D}^*(i,j) = d(i,j) + \min_F \begin{Bmatrix} D(i-1,j) \\ D(i-1,j-1) \\ D(i,j-1) \end{Bmatrix} \qquad \textbf{2.35}$$

The local constraint provides equal normalisation weight to horizontal, diagonal and vertical local transitions. Following the determination of the optimal trajectories, the warped $\boldsymbol{X_n}$ trajectory is then asymmetrically warped to match the number of points to the original reference trajectory. From equation 2.30, the number of time intervals on this warped trajectory must be greater than or equal to the number of intervals on the $\boldsymbol{X_{ref}}$ trajectory. The mean of all the points of the new warped trajectory that correspond with the same point on the $\boldsymbol{X_{ref}}$ trajectory is used as the new

point on the asymmetrically synchronised trajectory. A disadvantage of using the mean is that it may distort the relationship between the variables at this point.

2.4.3 On-line Dynamic Time Warping

The dynamic time warping algorithm above is not sufficient for on-line DTW, as the final endpoint constraint is not known as the batch trajectory is incomplete (i.e. the process is still ongoing). Kassidas et al. (1998) made adjustments to the algorithm in order to perform on-line DTW. Instead of a fixed final endpoint, one would have a set of final "endpoints" corresponding to the range of points for the reference trajectory on the grid that could possibly match the newest measurement $\boldsymbol{X_{new}} \in \mathbb{R}^{t \times J}$ of the incoming trajectory (figure 2.7). The range of points is defined by the global constraint. It follows that the size of the grid would have the dimensions of the number of intervals in the new trajectory, and the number of intervals of the reference trajectory up to the upper range limit imposed by the global constraint.

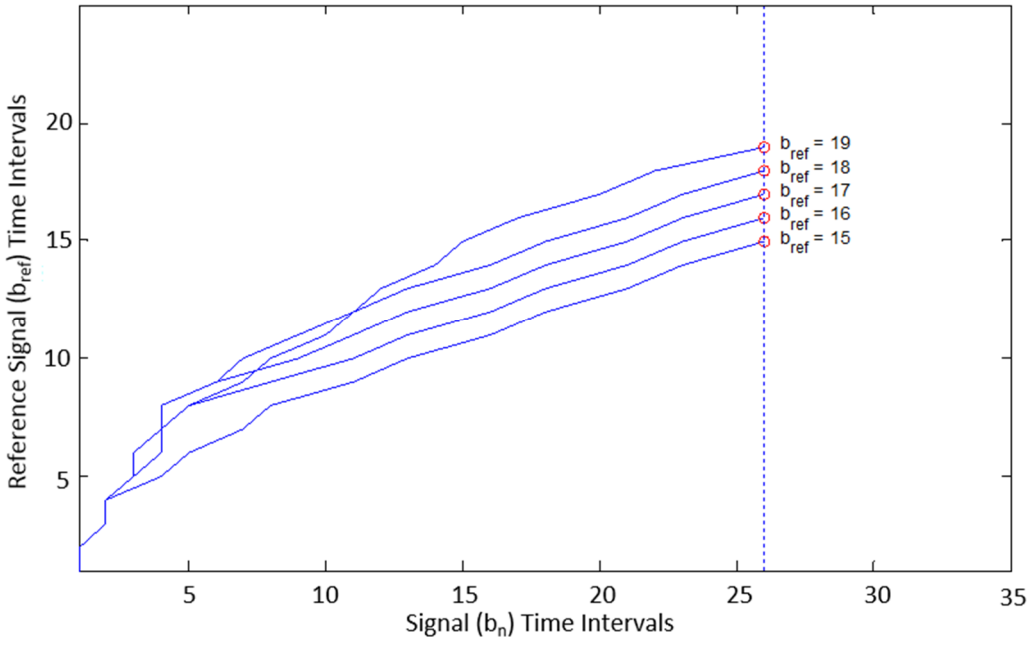


Figure 2.8: Potential "endpoints" for on-line DTW

One would then have a set of accumulated distances for each of the possible points. The point for the reference trajectory that corresponds to the minimum accumulated distance within this set is chosen. The data are synchronised up to this point. The missing data up until the end of the batch can be assumed in order to use multivariate statistical methods given by Nomikos and MacGregor (1995). This algorithm must be repeated every time a new measurement is available.

In off-line implementation, the normalisation factor of equation 2.32 was ignored as the reference trajectory was always the same length. Kassidas et al. (1998) showed that the normalisation factor based on a weighting function for symmetric warping could reduce to

$$N(w) = \sum_{k=1}^{K} w(k) = K_{ref} + K_n \qquad \textbf{2.36}$$

This normalisation factor is appropriate if the diagonal local transitions are given a weight of 2 to provide independence of the final distance to the number of path points (a diagonal is seen as one step horizontal step and one vertical step). However, the slight modification of the weights shown in equation 2.35 provides a weight of 1, equal to that of a horizontal or vertical local transition. Changing this weight makes the normalisation factor dependent of the path because diagonal transitions are favoured. This was justified by Kassidas et al. (1998) because it would result in smaller distortions of the reference and signal time axes, and the optimal warping path is of interest rather than the final cumulated distance itself.

In on-line implementation, the number of reference points changes depending on the endpoint chosen. Thus, for each endpoint the normalisation factor as defined by equation 2.33 should be different and it cannot be ignored when determining the minimum accumulated distances of each endpoint. Kassidas et al. (1998) does not address this explicitly for on-line implementation, so it is assumed that the normalisation factor was ignored.

The DTW algorithm uses a backwards search, meaning that a new path is created every time a new point is available which may differ from the path created for the previous time point. This could create some uncertainty during monitoring, as statistics for previous time points may give different results.

Computational Efficiency

A major drawback of DTW is the computational cost (Brown & Rabiner, 1982). Computational cost is defined as the time taken to make the required computations. In DTW, this is high because all the potential paths that can be followed have to be computed. This may be sufficient for off-line implementation, where the delay due to computational time does not have a meaningful effect on the monitoring as the batch has already been completed. However, this cost may have a considerable effect on on-line monitoring, as the synchronisation and subsequent analysis needs to be done quickly enough such that adjustments can be made to the process if necessary.

2.4.4 Relaxed Greedy Time Warping (RGTW) Approach

González-Martínez et al. (2011) proposed a time warping algorithm based on a relaxed greedy strategy for real-time monitoring. Training of the model is done off-line using conventional DTW without global constraints on a set of NOC batches, resulting in a number of optimal paths (one for each batch). Upper and lower limits are chosen as the maximum and minimum value for each point along the grid:

$$u(\boldsymbol{F}) = \max(\boldsymbol{f}_{i(1)}, \dots, \boldsymbol{f}_{i(K_{warp})}) \quad \textbf{2.37}$$

$$l(\boldsymbol{F}) = \min(\boldsymbol{f}_{i(1)}, \dots, \boldsymbol{f}_{i(K_{warp})}) \quad \textbf{2.38}$$

where $\boldsymbol{f}_{\boldsymbol{k}}$ is a column vector of the optimum path matrix $\boldsymbol{F} \in \mathbb{R}^{N \times K_{warp}}$. The maximum and minimum value for every n-th column on $\boldsymbol{F}$. These upper and lower boundaries are used to replace the global constraints in the conventional DTW algorithm (figure 2.9)

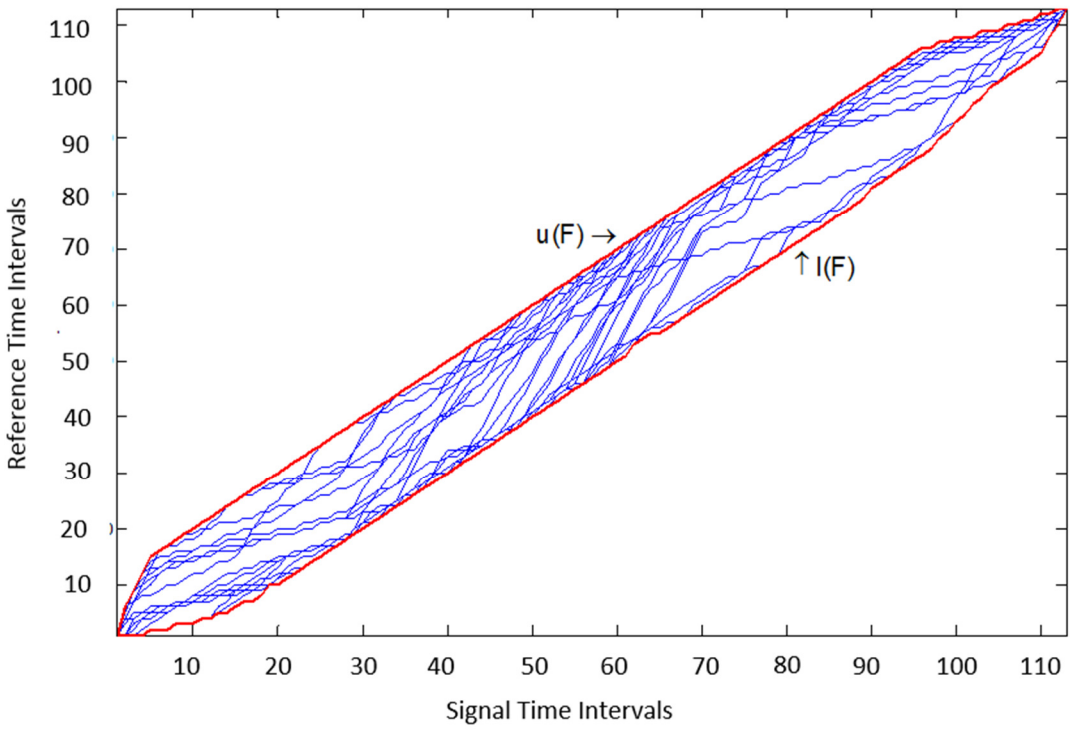


Figure 2.9: Upper and lower limits of the training data used as global constraints

Additionally, González-Martínez et al. (2011) addressed the issue of updating the boundaries to account for cases where the duration of the batches are longer or shorter than the NOC batches used to define the boundaries. If the endpoint at signal time k lies on one extreme of the boundary, the limit is updated. If it lies on the upper boundary, the upper limit is extended by 2 path points

(or the maximum reference point). If it lies on the lower limit, the next two signal points are given the same value as the limit at the current time t, and the points thereafter are lower limits is dropped by the difference between the limit at t and the limit at $t+2$:

$$u_i = \min\left(u_i + 2, K_{ref}\right), \qquad i = k+1, \dots, max(K_n) \tag{2.39}$$

$$l_i = \begin{cases} l_t & i = \{t+1, t+2\} \\ l_i - (l_{t+2} - l_t) & i = t+3, \dots, max(K_n) \end{cases} \tag{2.40}$$

The RGTW approach (figure 2.10) uses DTW to calculate the optimum paths within a certain sliding window. The optimal paths within this window are calculated every time a new set of measurements are available. The points outside of this window are fixed, and the endpoint constraint defined as the latest fixed point outside of this window. Once the paths have been assessed, the starting point of the new window will move one position forward, with the endpoint constraint as point that has just moved outside the window (which is now fixed).

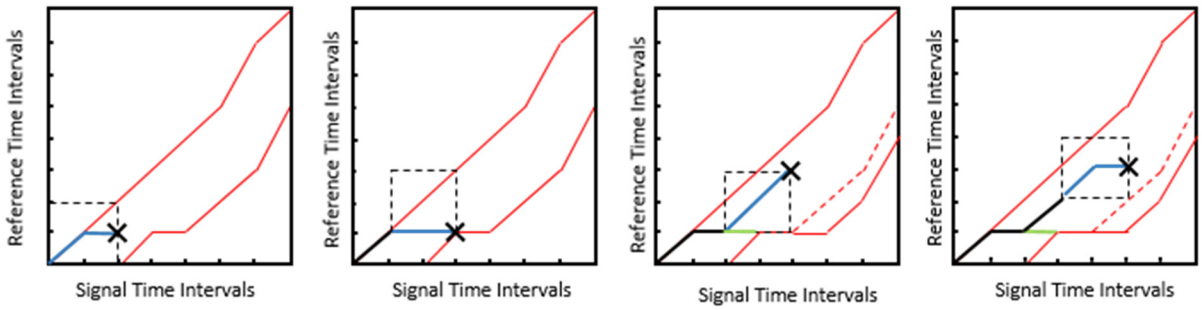


Figure 2.10: Visualisation of the RGTW approach. The red lines are the boundary constraints, the black line is the fixed synchronisation path, the blue line is the synchronisation path within the window (dashed box) and the green line is a discarded path (redrawn from González-Martínez et al., 2011)

González-Martínez et al. (2011) determined the window size using a cross-validation procedure. First, a window size was chosen. Leave one out cross validation was done on a set of NOC data in an offline manner, using one trajectory as the test batch and the remaining trajectories as the training batch (repeated for every batch). The test batch was synchronised in an on-line manner and compared to the same batch that was synchronised using DTW using the Pearson's correlation coefficient (PCC). The similarity index (δ) is found by:

$$\delta = \prod_{i}^{I} PCC\left(\boldsymbol{f}_{i,DTW}, \boldsymbol{f}_{i,RGTW}\right) \tag{2.41}$$

where $PCC(\boldsymbol{f}_{i,DTW}, \boldsymbol{f}_{i,RGTW})$ is the PCC between the synchronised paths found by the DTW and RGTW algorithms. This was repeated for a number of window sizes. An Analysis of Variance (ANOVA) was performed on the Fischer z-transformed correlation coefficients to test for statistically significant differences between window widths. For the p-values < 0.05, the Least Significant Difference (LSD) intervals were used to test for differences between the individual window widths.

González-Martínez et al. (2011) compared the results of the RGTW algorithm with those of the on-line algorithm given by Kassidas et al. (1998). Both algorithms were applied 1000 times to NOC data of a yeast fermentation simulation and the runtime was measured for each. The RGTW approach reported average running times of 0.241, 0.251 and 0.262 seconds for window widths of 1, 3, and 5 time intervals (the total time for the batch was 200-250 time intervals) compared with an average time of 1.935 seconds for the DTW algorithm. This is a considerable decrease in computational time. Computations were done on computer equipped with an Intel Core Duo with 4GB of RAM.

The performance of both RGTW and DTW for fault detection was also assessed. The authors found a noteworthy reduction in false alarms when using the RGTW approach compared with the DTW approach. These false alarms were attributed to the DTW resampling the optimum path at every point, which may be different for every sample. Conversely, the RGTW algorithm produces a synchronised sample after the first few steps, corresponding to window width. Further monitoring using RGTW was done on abnormal batches, and the SPE control charts were able to signal the faults from the start of the batch.

This approach is computationally faster when applied on-line at the expense of finding the global optimum path. However, the window selection procedure ensures a check on the performance of the algorithm compared with DTW. One drawback is that training the data using DTW without global constraints is still required, which may be computationally expensive.

The global constraints were excluded when synchronising the training batches such that the optimum synchronisation path would be found. However, Ratanamahatana & Keogh (2005) stated that aside from reducing computational time, global constraints were included to prevent pathological warping (where a relatively small section of a trajectory maps to a much larger section of another).

2.4.5 Hybrid Derivative Dynamic Time Warping (HDDTW)

HDDTW is another method proposed by Gins et al. (2012) to improve prediction accuracy and stability. Derivative time warping replaces the distance measure $[\boldsymbol{x}_{n,i(k)} - \boldsymbol{x}_{ref,j(k)}]$ in equation 2.33 with the derivative with respect to time. HDDTW approximates the measurements using piece-wise linear approximations due to the numerical instability of computing derivatives with noisy data. The approximation is compared to the actual signal and if the distance between the two grows too large (Gins et al. (2012) used the criteria of 3 times the standard deviation of the measurement error), a new linear part will be constructed to approximate the trajectory. It follows that fewer linear parts need to be constructed in places where the trajectory is relatively constant, and more when the trajectory has features. This is advantageous during constant trajectory periods compared to the original on-line DTW as the DTW has trouble mapping the correct values due to noise (figure 2.11).

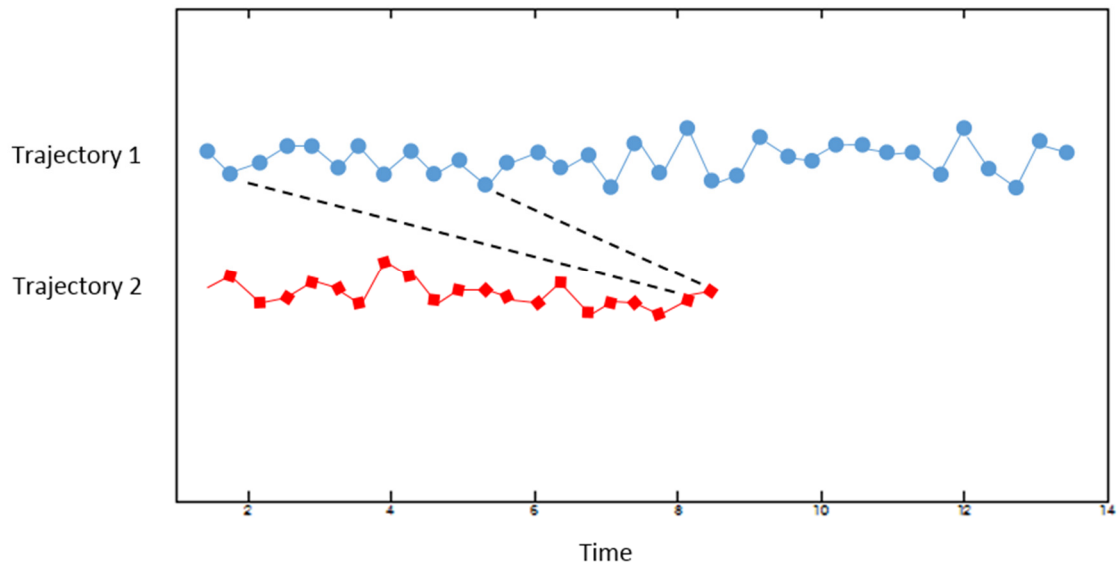


Figure 2.11: Potential incorrect mapping of points using conventional DTW. Redrawn from Gins et al. (2012)

The approach by Gins et al. (2012) was to use the Kassidas et al. (1998) algorithm during feature rich zones, and the HDDTW algorithm during feature poor (constant) zones. Furthermore, an aspect of the accumulated distance was noted. If a reconstructed path intersects with a previous path, it must follow the same path back to the origin from the point of intersection. If this occurs, the rest of the path is immediately known and no further computation has to be done. This saves on computation time.

No direct comparison of DTW was done, but computational expense was shown not to delay the near-real time synchronisation in the industrial batch polymerisation process studied by

Gins et al. (2012). The authors reported a final quality estimate updated in approximately 0.2s (using MATLAB 7.8 on a computer equipped with an Intel Core2 Duo Quad Q9550 2.83 GHz processor and 2×2 GB RAM). Expert knowledge is also required to determine the global constraints of the system, similar to DTW. It should also be noted that the performance around some of the phase transitions was such that severe distortions in the model predictions could occur, and predictions were not made during these phase transitions.

2.4.6 Correlation Optimised Warping (COW)

COW was proposed as an alternative to DTW by Nielson et al. (1998) for the use of peak detection in chromatographic profiles. The method aligns two signals by piecewise linear stretching and compression (Pravdova, et al., 2002). The optimal alignment is determined by the correlation of the aligned fragments of signals. Instead of the distance function used in DTW (equation 2.33), the correlation coefficient between corresponding parts on the target signal and reference signal are used:

$$corr(k) = \frac{(\boldsymbol{v}_{\boldsymbol{b}_{ref}} - \bar{v}_{b_{ref}})^T \cdot (\boldsymbol{v}_{\boldsymbol{b}_n}{}' - \bar{v}_{b_n}{}')}{std\left(\boldsymbol{v}_{\boldsymbol{b}_{ref}}\right) \cdot std(\boldsymbol{v}_{\boldsymbol{b}_n}{}')} \qquad \textbf{2.42}$$

where $\boldsymbol{v}_{\boldsymbol{b}_{ref}}$ is a segment of the reference trajectory and $\boldsymbol{v}_{\boldsymbol{b}_n}{}'$ is a segment of the trajectory that has been warped by linear interpolation to get the same length as the reference trajectory, and k is the point on the grid. The standard deviation and mean are denoted by $std(\boldsymbol{v})$ and $\bar{v}$ respectively. The cumulative benefit is calculated based on these values and the highest benefit functions are kept, indicating the highest correlation.

Tomasi et al. (2004) compared the two algorithms (within a chromatography framework), and reasoned that COW may be regarded as a case of DTW where additional constraints are added to reduce the search space for optimal warping and the correlation coefficient is employed as optimisation criterion. They concluded through their case study (with chromatographic data) that the performance of COW was very similar to that of DTW with tight constraints. Some artefacts due to warping were found at the peaks of the data in both cases. However, the application to batch trajectories may differ due to the different nature of the data. Tighter constraints may not find the optimal path when using DTW/COW as a pattern recognition tool for batch data, but perform well when analysing chromatographic data, where preserving the shape of the peaks is important. No comparison of computational expense was reported on, but a general mention was made that the COW algorithm is faster and less memory-demanding than DTW due to the need of the added flexibility of DTW, resulting in more paths that must be computed.

Fransson et al. (2006) applied the COW algorithm to batch data, resulting in satisfactory alignment. However, no direct comparison between COW and DTW was done. It is also worth mentioning that the prediction model, and consequently the synchronisation procedure was only done at every 11th data acquisition point (corresponding to an update every 5 minutes).

Table 2.1 summarises the approaches discussed above.

Table 2.1: Summary of Synchronisation Techniques

Method	Comparative between Trajectories	Caveats
Indicator Variable	Representative Variable	IV not always available
Dynamic Time Warping	Euclidean Distance	During featureless zones may warp to the wrong value, may produce false alarms when recalculating on-line trajectory
Relaxed-Greedy Time Warping	Euclidean Distance within a window	Requires off-line DTW and correct window size choice
Hybrid Derivative DTW	Derivative with respect to time	Some phase transitions left out of MPLS, Expert knowledge required during phase transitions and constraint parameters
Correlation Optimised Warping	Correlation	Search space is small therefore may not find the optimum path

The IV and DTW approaches have been used most extensively in the literature. Garcia-Alvarez et al. (2012) used the IV approach to align transient data for a sugar factory simulation case study and the DTW approach to align data from a laboratory plant case study. Rato et al. (2016) compared the IV and DTW approaches and found the DTW approach to be superior in the penicillin cultivation process simulation case study, but was detrimental for fault detection in the semi-batch reactor case study as it distorted the batch specific variations. Alternative approaches that unfolding have also been proposed (Ge & Song, 2014; Westad, Gidskehaug, Swarbrick, & Flåten, 2015), but these approaches are restricted to variable-wise unfolded data. The lack of versatility in these approaches make them inadequate for use in this thesis.

2.5 Review of Phase Division and Identification Techniques

Phase identification is necessary when applying different models to different phases yields better accuracy for monitoring and prediction purposes compared with a single model throughout the whole process. Kourti (2006) stated when defining synchronisation that the one goal that synchronisation aims to achieve is to establish common start points for different phases of the run. The synchronised trajectories will align the points of a trajectory onto a reference trajectory, thus establishing the common start points of the different phases, but will not provide information on how to divide the phases. The purpose of this section is to review the techniques in the literature that address this issue, specifically using MPLS models related to on-line fault detection and end-of-batch quality prediction.

Camacho et al. (2008) outlined three different approaches to fit multi-stage, multi-phase models:

- ***Expert knowledge***. Phases are defined from expert knowledge about the process, and separate models are used to represent each phase. These phase changes usually correspond to changes in operational phase or stage (Yao & Gao, 2009). The locations of the phase division points are found using prior knowledge of the process. Expert knowledge was used to divide phases and apply different models by Ündey et al. (2003).
- ***Process analysis***. Phases are defined from features of the process after analysis. Again, these phases usually correspond to changes in operational phase or stage (Yao & Gao, 2009). Prior knowledge regarding the variables that contain the features associated with phase changes is required. Doan & Srinivasan (2008) used singular points (sharp changes, extrema, trend changes) in some key variables to identify the phase changes.
- ***Data-based automatic methods***. Phases are defined from analysis of the correlation structure of the variables. A significant change in correlation structure is indicative that separate models may be needed to model the data effectively, and phases are defined as such. These correlation phases do not necessarily correspond to changes in operational phase or stage. Additionally, these methods do not rely on prior process knowledge and as a result can be applied to different batch processes. Lu & Gao (2005) proposed stage-based PLS modelling, which uses a clustering algorithm to group similar correlation structures (section 2.5.2), whereas Camacho et al. (2008) proposed the multiphase (MP) algorithm that uses a greedy optimisation approach to find the best division of phases (section 2.5.3).

A considerable advantage of using data-based methods is that the phases are divided directly according to correlation structure, which ensures that different models are used only when necessary. However, as with most data-based methods, certain criteria need to be defined for the

algorithms to satisfy in order to be effective. Additionally, if the correlation phase boundaries do not match easily-identifiable physical changes in the process variables or operational phase/stage boundaries, on-line phase identification may be difficult to achieve. Accurate synchronisation is required to align the trajectories so that the correct model is applied to the data.

2.5.1 Singular Points

Qian and Srinivasan (2005) noted that the majority of features of the signal are concentrated in a small number of points. These points are referred to as singular points (SPs) and include points where sharp changes occur, local extrema points and trend change points (figure 2.12). The points can be identified using the robust (to account for noise) first and second derivatives of the variables. Doan and Srinivasan (2008) used these points to separate the batch trajectory into multiple phases in real-time. The ability of SPs to be identified in real-time allows for both off-line and on-line implementations. On-line implementation first involves generating a phase-based reference model from historical data, then comparing the incoming data to the reference model. A generalised version of the algorithm given by Doan and Srinivasan (2008) for generating the reference model is as follows:

1) Identify one of the batches in the historical database that is representative of the average run, denoted as the "golden batch".
2) Select one or more key variables that reflect the important phases and phase changes.
3) Identify the SPs in the profiles of the key variables in the golden batch using an SP algorithm and shortlist the SPs that correspond to phase changes.
4) Segment the batch trajectory into phases based on the time occurrence of the relevant SPs.
5) Identify corresponding SPs of the other historical batches.
6) Synchronise each historical batch with the golden batch using time warping.
7) Generate a statistical model for each phase.

Generating such a reference model requires prior knowledge of the process when selecting the key variables that reflect important changes in the process and shortlisting the SPs that correspond to phase changes. In order to assess phase changes in on-line monitoring, new measurements are assessed using the SP algorithm. SPs that correspond to phase changes in the reference model are indicative of a phase change in the new data.

In order to synchronise the trajectories for the off-line case, conventional DTW was used. However, for on-line synchronisation, the Qian and Srinivasan (2005) used a modified version of the DTW algorithm based on a greedy search. This method, originally proposed by Qian and Srinivasan (2005), significantly reduces computational time compared to DTW at the expense of

potentially not finding the globally optimal warping. Qian and Srinivasan (2005) argued that using this time warping method in conjunction with SPs is sufficient because the latter incorporates the information about the key points in the process.

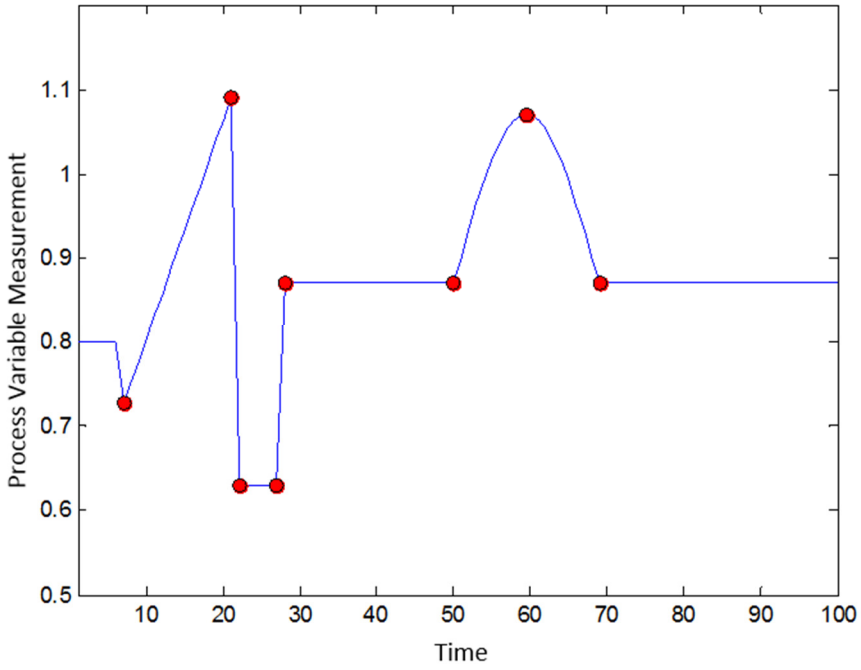


Figure 2.12: Batch trajectory showing singular points (SPs). Redrawn from Qian and Srinivasan (2005)

2.5.2 Stage-based PLS Modelling

A data-based approach was used by Lu and Gao (2005) in order to separate phases with different correlation structures automatically. The approach is an extension of the sub-PCA method proposed by Lu et al. (2004). It should be noted that the stages referred to by Lu and Gao (2005) are not the same as the operational stages defined in this thesis – rather these stages correspond with the definition of correlation phases. The principle behind stage-based PLS modelling is that major changes in the regression parameter matrix $\boldsymbol{\beta}$ from equation 2.19 will indicate a change in correlation phase, as this matrix contains the correlation information. Correlation phases are divided using a k-means clustering algorithm, where the time intervals are separated into a number of clusters based on the cluster with the nearest mean. The procedure for identifying the correlation phases is done off-line as follows:

1) The data are synchronised, centred and scaled.
2) A PLS model is built for every one of the K time intervals of the incoming data. Each model relates the variables at that particular time-slice k to the quality variables at the end of the batch run.

3) A k-means clustering algorithm is performed on the regression parameter matrix to group the time intervals with similar correlation structures.
4) Correlation phases are divided according to these clusters along with certain thresholds.
5) Process analysis is done to determine the critical-to-quality correlation phases and models are generated for each correlation phase. The method is described by Lu and Gao (2005).

In step 3, the initial number of clusters have to be specified, and clusters are merged if the distances between the centres are below a certain threshold. The choice of threshold offers a balance between complexity and accuracy – a smaller threshold would result in more accurate modelling, but more clusters; whereas a larger threshold would reduce the overall number of clusters, but reduce the modelling accuracy for each cluster. It was also noted by Camacho et al. (2007) that the clustering algorithm does not take into account sampling time ordering. This may result in clusters that contain non-consecutive sampling times (figure 2.13). No automatic post-processing to correct this was proposed.

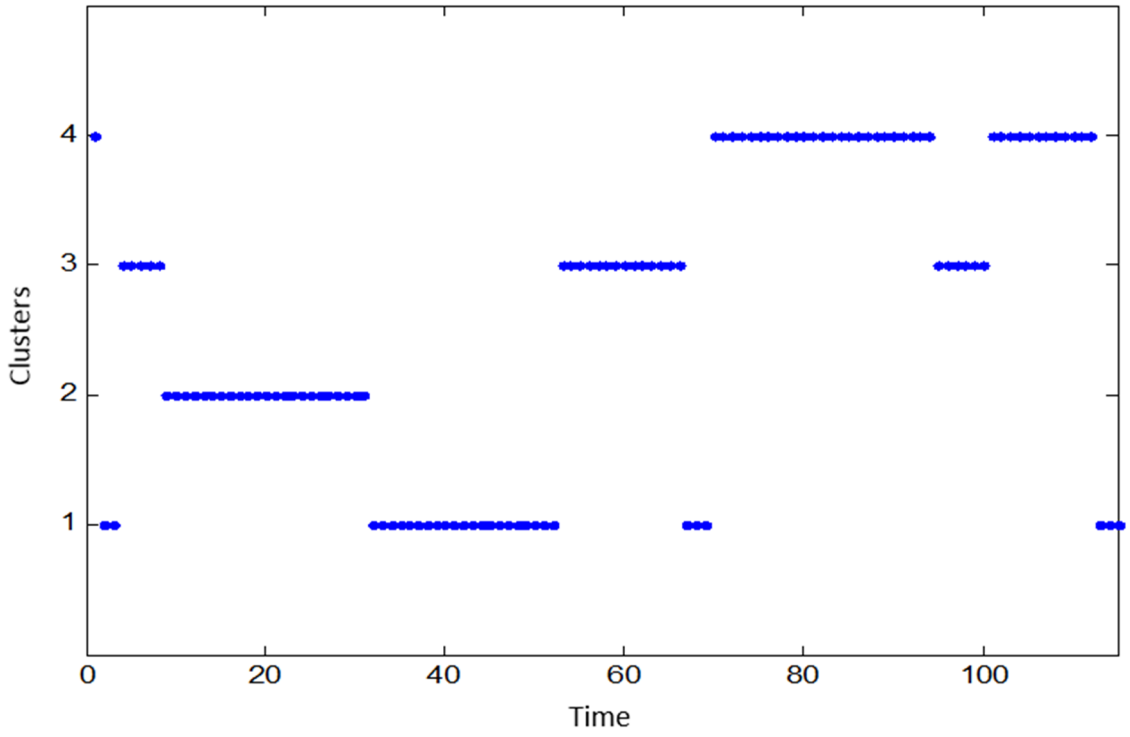


Figure 2.13: Clusters identified after applying clustering algorithm showing non-consecutive clusters. Redrawn from Camacho & Picó (2006)

The models generated are PLS models which can be applied to the two-way data at each time step, therefore no unfolding is necessary. The regression parameter matrix for each correlation phase p is found by taking summing the K_p regression parameter matrixes over the phase and dividing by the number of time intervals in the phase. This requires the length of the phase to be known, and

in batches with uneven length, the data need to be synchronised. In the injection molding case study used by Lu and Gao (2005), incoming batch data were aligned using an indicator variable.

For on-line application, synchronisation is done to the incoming data to identify the relevant phase of the variables. Once the phase is identified, the relevant model can be applied. The process analysis method determines what information the model uses – it can only use the information of the relevant phase, or it can use the information of more than one phase.

Accurate synchronisation is important for on-line phase identification in this case, as the clustering algorithm may not divide the phases exactly based on any expert rules or changes in process variable trajectories. Instead of synchronising the trajectories, Lu et al. (2004) used the monitoring statistics (Hotellings T^2 and SPE) to determine if the phase should change when applying their PCA-based modelling strategy to the same injection molding case study. If the monitoring statistics exceeded the control limits, the model for the next phase was applied to the same data. If the statistics fell below their limits in this next phase, the correlation phase was identified as the new phase. This method may also be appropriate when applying PLS models to the two-way data as the only input data needed are the data at the current time. However, if MPLS is applied, proper synchronisation is needed. For batch-wise unfolding, the future data added to fill the matrix may bias the statistics in some way and accurate phase identification may not be feasible.

2.5.3 Multiphase Partial Least Squares (MPPLS)

Another data-based approach was proposed by Camacho et al. (2008) to identify correlation phases in batch trajectories automatically, called the multiphase algorithm. The algorithm is a generalised version of one that was originally proposed for PCA (Camacho & Picó, 2006). It consists of a high-level routine that sets the number of phases, and a low-level routine that identifies the division in phases of the sampling time mode. The routines as laid out by Camacho & Picó (2006) are as follows:

High Level Routine

1) Select an unfolding and a modelling method
2) Fit model *M* with a_{ini} Principal Components (PCs)
3) Repeat while adding PCs to *M*
4) a) Add a PC to the model *M*, obtaining model M_1. Compute $\delta(M_1, M)$ from:

$$\delta(M_1, M) = \frac{QPE(M) - QPE(M_i)}{QPE(M_0)} \qquad \textbf{2.43}$$

where QPE is the Quadratic Prediction Error, and M_0 represents the pre-processing information – the averages and weights of the variables.

b) Divide the Model M in two sub-models with the low-level routine, obtaining Model M_2. Compute $\gamma\big(\delta(M_1, M)\big)$ from:

$$\gamma\big(\delta(M_1, M)\big) = k_\gamma \cdot \delta(M_1, M) \tag{2.44}$$

c) If $\delta(M_1, M) > \gamma\big(\delta(M_2, M)\big)$ and $\delta(M_1, M) > T$, Replace M with M_1.

d) If $\delta(M_1, M) \leq \gamma\big(\delta(M_2, M)\big)$ and $\delta(M_1, M) > T$, Replace M with M_2 and compute the repeat loop (4) recursively for both sub-models.

Low-Level Routine

1) For each sampling time (k) in the current interval modelled by M
   a) If the subdivision in k generates two segments in length higher than the minimum established ($minL$):
      i) Fit model for the two segments
      ii) Compute β^k from:

$$\beta^k = QRE - QRE^k \tag{2.45}$$

where QRE is the Quadratic Residual Error of model M and QRE^k is the Quadratic Residual Error Obtained when M is divided into two sub-models in k.

      iii) If β^k is highest to the moment $(\beta^k = max_t\, \beta^t : t = minL, \ldots, k)$, update better subdivision.

The high level-routine requires synchronised, scaled and centred data. Any unfolding method can be chosen. It uses a greedy optimisation strategy to determine the number of phases. For both routines, a number of improvement thresholds are required to determine:

- Whether to add a new latent variable to the model
- Whether to divide the model into separate phases
- If the overall improvement of model accuracy is justified by the increase complexity of the model

Camacho et al. (2007) provides many additional algorithms that can be used for these thresholds. Additionally, the algorithm can be run for a number of different unfolding methods and improvement parameters, and a merging algorithm can be used to determine the optimal model for each phase. This makes it possible for each phase to have a different unfolding method that best describes the performance of the model.

The QPE for MPPLS may be calculated as the Predicted Residual Error Sum of Squares (PRESS) using cross-validation techniques. However, because the low-level routine must fit a model for the two segments for each value of k, Camacho et al. (2007) recommended using the residual error (QRE) as opposed to the prediction error (QPE) to ease the computational burden of using cross validation for every time k. Calculating the residual error does not use cross-validation - rather the sum of the squared difference between the actual values and the values expected by the model.

The approach is robust, but complex as it requires many thresholds to be chosen, which may require some knowledge of the nature of the data. The greedy nature of the phase division strategy may also result in a considerably different phase division when different parameters are chosen such as number of starting latent variables, minimum division length ($minL$) etc. The algorithms proposed to deal with these thresholds increase the complexity further. However, the approach incorporates sampling time ordering, which is advantageous compared with stage-based PLS modelling.

2.5.4 Phase Identification using Time Warping

In the time warping procedures (sections 2.4.2 – 2.4.6), incoming data are synchronised to a reference trajectory. Generally, the phase division indexes of the reference trajectory are known, as the procedure is applied off-line. During on-line application, the incoming data ($\boldsymbol{x}_{new}$) is synchronised to the reference trajectory. The current index of the incoming data will therefore correspond to an index of the reference trajectory. This index can be compared to the known phase boundaries and the current phase can be identified. The method is easy to implement on-line as synchronisation must be done in any case, but inaccurate warping might not result in the correct phase being identified.

The table on the next page summarises the phase division and phase identification approaches discussed in this section.

Table 2.2: Summary of Phase Division and Phase Identification Techniques

Method	Type	Caveats
Rule-based	Expert Knowledge	Requires expert knowledge of the process, restricted only to known phases
Singular Points	Process Analysis	Requires known features
Stage-based PLS	Data	Time not explicit, therefore post-processing required
MPPLS	Data	Complicated algorithm, accumulated phase effects not considered
Time Warping		Inaccurate warping may cause inaccurate phase identification

CHAPTER 3 Methodology

In the previous chapter, techniques used for on-line fault detection and end-of-batch quality prediction were reviewed. This section presents the techniques that will be used in this thesis, the procedure for analysis including the metrics used and the case studies selected for application of the techniques. Section 3.1 presents the techniques used from chapter 2. Section 3.2 describes the different procedures used: off-line training and on-line application. Section 3.3 goes into specific detail about the algorithms used, section 3.4 outlines the metrics used and fault detection and end-of-batch quality prediction methodology and section 3.5 introduces the case studies selected for application of the methods.

3.1 Techniques Used in this Thesis

The following section outlines the techniques that will be used in this thesis. A summary of the techniques is presented in table 3.1:

Table 3.1: Summary of Techniques used in the Thesis

Step	Off-line/On-line	Technique	Reference Section
Synchronisation	Off-line	Off-line DTW	2.4.2
	On-line	RGTW	2.4.4
Phase Division	Off-line	Expert Rules	2.5
		MPPLS	2.5.3
Phase Identification	On-line	On-line Synchronisation	2.5.4
Unfolding	Both	Batch-wise	2.3.3
Scaling	Both	Auto Scaling	2.3.4
Model Training	Off-line	MPLS	2.2.2
Future Data Imputation	On-line	Zero deviations	2.3.5

Multivariate statistical methods on two-way data have been shown to be effective methods to model and monitor batch processes. This thesis deals specifically with end-of-batch prediction, therefore MPLS models will be applied (Nomikos and MacGregor, 1995). In MPLS, PLS models (section 2.2.2) try to explain the variance of batch-wise unfolded input data ($\mathbf{X}_{BW}$, section 2.3.3) that are more predictive for the product quality ($\mathbf{Y}$). Before the models can be trained on the data, they must be auto scaled (section 2.3.4). Global models were used by Nomikos and MacGregor (1995), but separate MPLS models can be used for each distinct phase if a phase-wise modelling approach is performed (2.3.1).

Off-line pre-processing of batch data is necessary before models can be trained. Off-line dynamic time warping (section 2.4.2) was performed on slab scaled input data in order to align their trajectories. The procedure for off-line synchronisation is presented in Appendix F.1.3.

The accuracy of using phase-based models versus global models is also investigated. Off-line phase division is achieved by dividing the phases according to known phase boundaries (i.e. using expert knowledge, section 2.5) or using an automatic algorithm to divide the data based on correlation structure changes of the data over time. The Multiphase Partial Least Squares (MPPLS) algorithm (section 2.4.5) will be implemented and used to find these changes in correlation structure.

For on-line application, pre-processing needs to be performed. The steps performed are synchronisation, phase identification and future data infilling. On-line dynamic time warping (section 2.4.3) is an effective and robust method to synchronise batch trajectories to the training data set, however the computational time is a concern. The relaxed greedy time warping approach (RGTW, section 2.4.4) is a compromise that reduces the computational time and returns results that can be compared to conventional on-line DTW. An added advantage of using RGTW is that the window size can be adjusted. Setting the window size larger than the length of the largest batch results in an on-line DTW approach with the only difference being the way that the search space limits are defined. RGTW will be implemented for on-line synchronisation.

On-line phase identification is needed when phase-wise models are trained in order for the correct model to be applied to the correct phase. Phase identification is performed in this thesis using on-line synchronisation (section 2.5.4). The data are synchronised to a reference batch, and the time index of the reference batch that was synchronised to the current time index of the new batch will be used to determine the current phase, as the phase boundaries of the reference batch are known.

In order to apply the model to the data, the future data have to be assumed. Zero deviations will be used i.e. the future data will be added to the scaled, centred matrix as zeroes (section 2.3.5). This method is simple to perform and has been shown to be one of the more accurate approaches. The table below summarises the techniques used in this thesis for the different aspects of the routines.

3.2 Off-line Training and On-line Application Procedures

This section describes how the models were trained and applied to the new data, including all the pre-processing steps that were employed. A detailed description of the procedures is given in Appendix F. The procedure consists of an off-line routine and an on-line routine, shown in figure 3.1. The purpose of the off-line routine is to train the models. In this routine, certain pre-processing and modelling results were stored and used in the on-line routine. These results are configured for on-line use during the set up routine. During the on-line routine, new data are pre-processed and models are applied to generate the monitoring statistics and end-of-batch quality predictions.

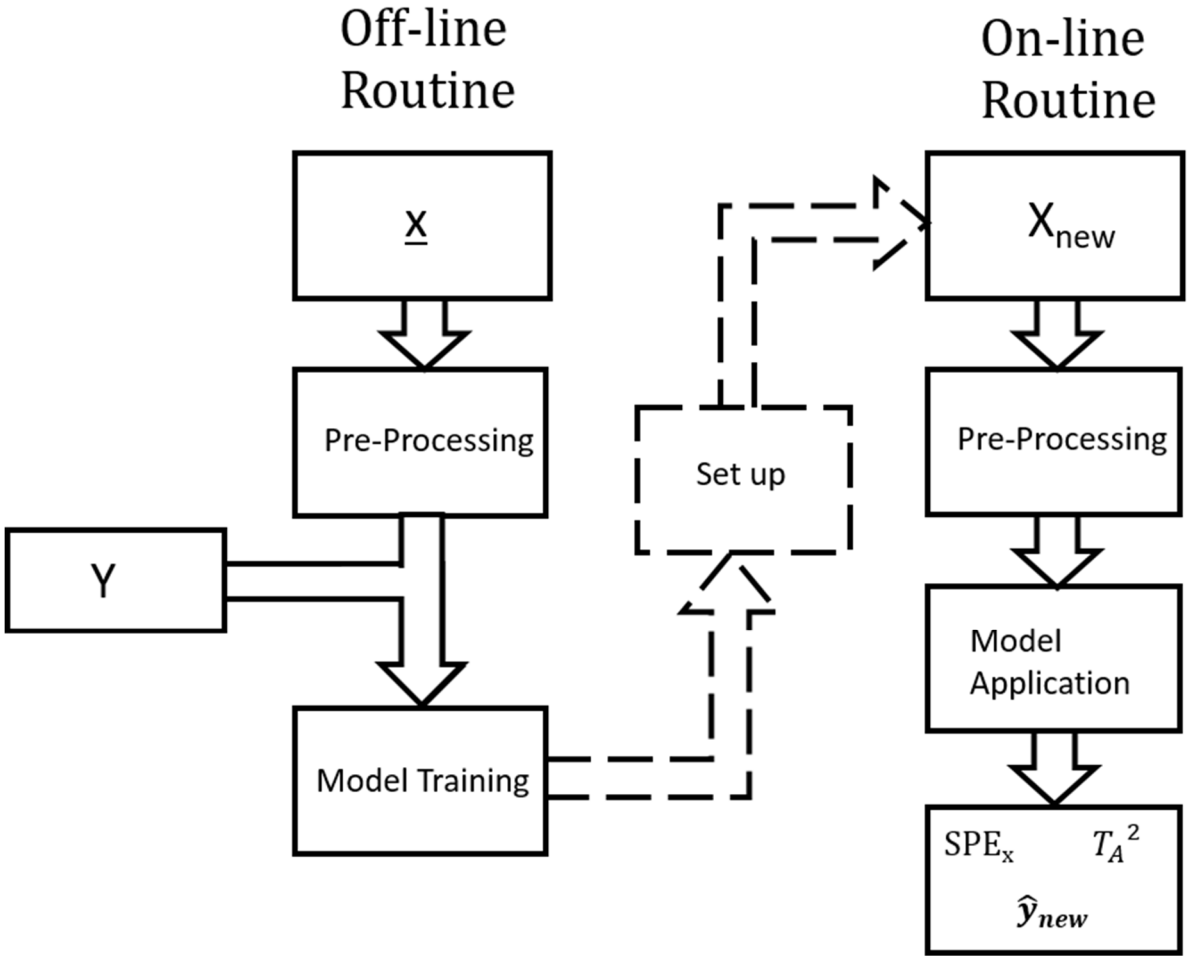


Figure 3.1: Overview of Procedures

3.2.1 Off-line Training

Before any statistical model can be applied, models have to be trained on NOC data. First, the data are pre-processed, then models are trained on the pre-processed data.

Pre-processing

Figure 3.2 shows a flow diagram of the pre-processing step. The steps are as follows:

1) Three-way NOC data are obtained for training
2) The data are split according to known operational phase boundaries, resulting in Π sets of data. This is done because the off-line DTW algorithm is computationally expensive, and synchronising the data per phase limits the search area, thereby reducing the computational time.
3) Slab scale each of the Π sets of data and store the means $(\mu_{\pi,slab})$ and standard deviations $(s_{\pi,slab})$ for each operational phase
4) Synchronise the data using off-line Dynamic Time Warping (Appendix F.1.3). The resulting matrixes will each have the same number of time indexes, $K_{\pi,ref}$. The (scaled) reference trajectory and the phase boundaries of the reference trajectory must also be stored.
5) Unscale the data using the means and standard deviations from step 3.
6) Concatenate the synchronised data such that the total number of time indexes is $K_{ref} = \sum_{\pi=1}^{\Pi} K_{\pi,ref}$
7) Auto scale the synchronised data and and store the means $(\mu_{\pi,auto})$ and standard deviations $(s_{\pi,auto})$
8) Divide the data according to phases in three different ways
    a. Globally i.e. one phase
    b. Using expert rules i.e. according to known phase boundaries
    c. Using the MPPLS algorithm (section 3.3.1) i.e. according to correlation phases

    The next step is performed for each set of phase divided data
9) Batch-wise unfold each phase of the data

The result is P sets of batch-wise unfolded matrixes for each type of phase division, where P is the number of phases from step 8. The slab scaling and auto scaling parameters, as well as the reference trajectory and its phase boundaries, must be stored for use in on-line application.

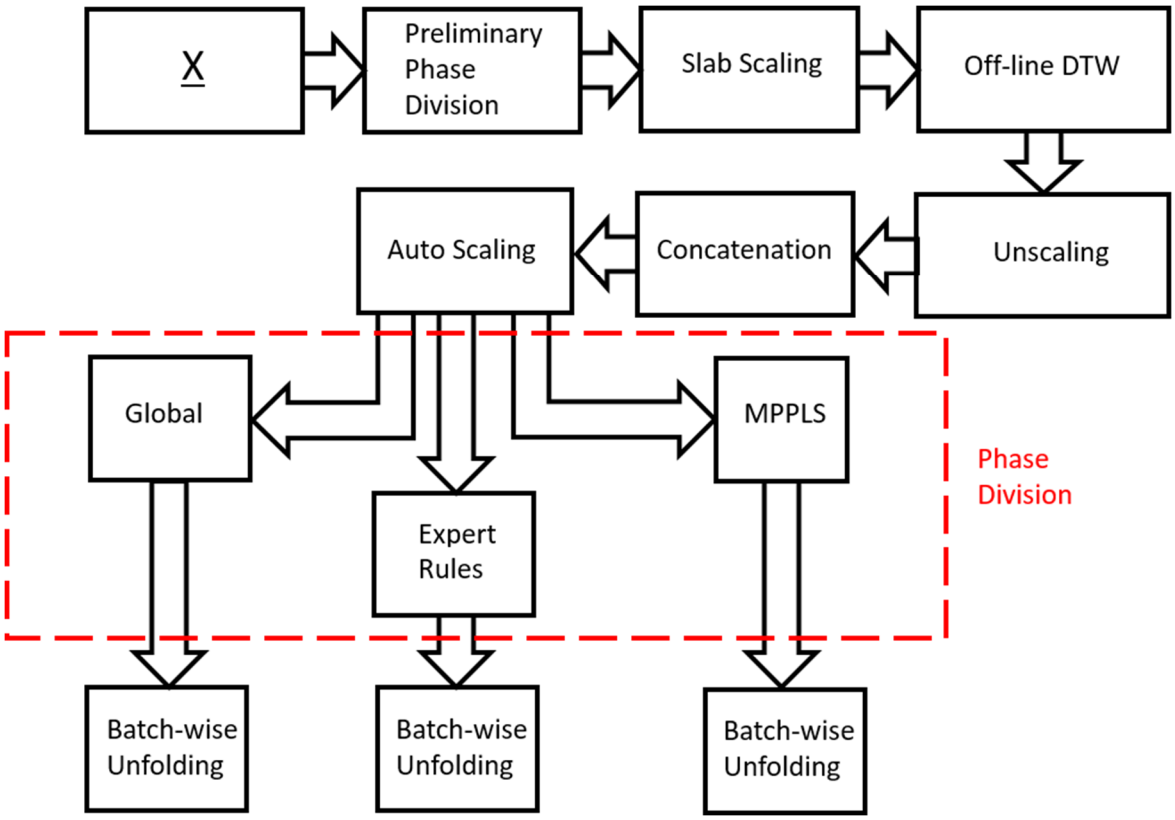


Figure 3.2: Off-line Pre-processing Procedure

Model Training

After the data have been pre-processed, models can be trained for each matrix. The resulting matrixes from the pre-processing are:

- One global batch-wise unfolded matrix
- Π sets of operational phase-divided, batch-wise unfolded matrixes
- P sets of correlation phase-divided, batch-wise unfolded matrixes

The product data, $\tilde{\boldsymbol{Y}}$ are the same for each matrix. Figure 3.3 shows the flow diagram of the model training routine. The steps are as follows:

1) The product data of the NOC training batches are obtained
2) Auto scale the product data and store the means and standard deviations
3) Select the number of retained principal components (PCs) using the MSE calculated by leave-one-out cross validation (LOOCV) and the variance explained in the product variable. For the case of the MPPLS phase division, the retained components are already known, therefore LOOCV does not have to be done. LOOCV is explained in the next section.

4) Build a PLS model using the SIMPLS algorithm and the selected number of principal components. The algorithm returns, amongst others, the x-scores matrix $\boldsymbol{T}$, the weights matrix $\boldsymbol{R}$ and the regression matrix $\boldsymbol{\beta}$. Store these matrixes for use in model application.
5) Calculate the Hotelling's T_A^2 parameters, s_{t_a} and $c = \frac{A(I^2-1)}{I(I-A)}$, as well as the limit, $F_{(A,I-A)_\alpha}$ at $\alpha = 0.05$
6) Calculate the SPE_x limits using equations 2.23 – 2.25 at $\alpha = 0.05$

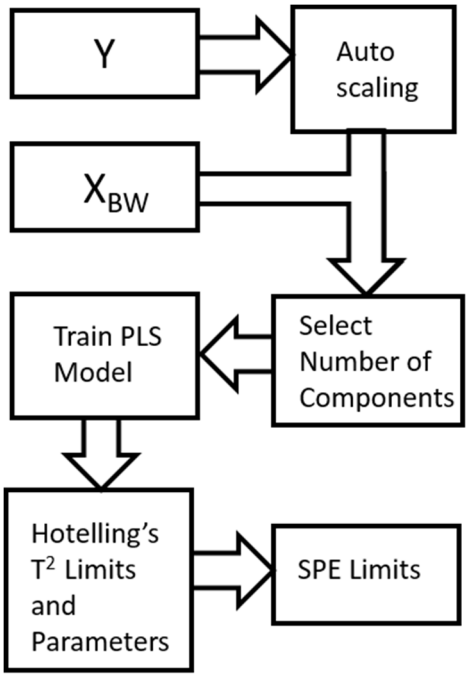


Figure 3.3: Off-line Model Training Procedure

Leave-one-out Cross Validation (LOOCV)

LOOCV was used to test the prediction power of the model. The mean errors resulting from LOOCV were used to select the number of principal components to use in the model.

For each batch of training data (denoted as test batch):

1) Exclude the test batch from the matrix to form the training matrix $\boldsymbol{X_{LOOCV}} \in \mathbb{R}^{(I-1)\times JK_{ref}}$
2) Fit a model to the data of the training matrix using PLS regression.
3) Predict the end quality of the test batch using equation 2.26 at every time interval k. Use data up until the k-th time index for each prediction, and fill the rest of the matrix with zeroes.

4) Find the mean squared prediction errors between the predicted and actual quality values.

$$MSE_i = \frac{\sum_{k=1}^{K}(y_i - \hat{y}_{ik})^2}{K} \quad \textbf{3.1}$$

Thereafter, find the mean squared error across all batches

$$MSE = \frac{\sum_{i=1}^{I} MSE_i}{I} \quad \textbf{3.2}$$

3.2.2 On-line Application

This section outlines the application of the models to the test data. First, the pre-processing and prediction information from off-line training must be configured for use in on-line training. Thereafter the incoming data are pre-processed and monitoring statistics and predictions are generated. A more detailed description of the methods are given in Appendix F.2.

Preliminary Set up

During on-line application, the phases are unknown and as a result on-line synchronisation must be done to the batch as a whole. The off-line DTW synchronised each batch separately, therefore the slab scaling parameters for each reference trajectory are operational phase-specific. The trajectories first had to be unscaled to their actual values, concatenated and slab scaled as an entire batch. A potential issue with concatenating the reference trajectories in this way is that each reference trajectory was chosen due to their correlation with the training data of that specific phase, and as a result the reference trajectory across the phase boundary will probably not be continuous, as shown in figure 3.4.

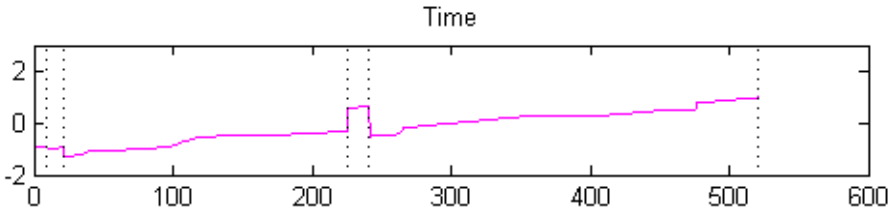


Figure 3.4: Jump discontinuities across phase boundaries in the reference batch

The following parameters also had to be configured:

1) The search space limits for on-line synchronisation;
2) The auto scaling parameters;
3) The prediction information.

Pre-processing

On-line pre-processing was done for every test batch, and repeated for different phase divisions and synchronisation parameters. A schematic of the process is shown in figure 3.4. Every time a new measurement $\tilde{\boldsymbol{x}}_{\boldsymbol{new}} \in \mathbb{R}^{1\times J}$ of a test batch is available, it was added to the existing matrix $\tilde{\boldsymbol{X}}_{\boldsymbol{prev}} \in \mathbb{R}^{(k-1)\times J}$ (forming $\tilde{\boldsymbol{X}}_{\boldsymbol{new}}$) and pre-processed as follows:

1) $\tilde{\boldsymbol{X}}_{\boldsymbol{new}}$ was slab scaled using the slab scaling parameters determined for the NOC training data during the preliminary setup routine.
2) The scaled dataset $\boldsymbol{X}_{\boldsymbol{new}} \in \mathbb{R}^{k\times J}$ was synchronised on-line using the RGTW algorithm (section 3.3.2) to form $\boldsymbol{X}_{\boldsymbol{sync}} \in \mathbb{R}^{k_{ref}\times J}$.
3) The current phase was identified by comparing the latest time index of the synchronised trajectory to the reference phase boundary time indices, determined off-line.
4) The entire synchronised dataset was unscaled using the synchronisation scaling parameters.
5) The entire synchronised trajectory was auto scaled using the scale parameters chosen for prediction.
6) The future data from $k_{ref}{:}K_{ref}$ were imputed as zeroes such that $\boldsymbol{X}_{\boldsymbol{sync}} \in \mathbb{R}^{K_{ref}\times J}$ where $K_{ref} \geq k_{ref}$.
7) The data from the time index of the current phase starting boundary to the current phase end boundary were isolated to form $\boldsymbol{X}_{\boldsymbol{p}} \in \mathbb{R}^{K_{p,ref}\times J}$, where $K_{p,ref}$ is the number of reference time indices within the current phase.
8) The data were unfolded batch-wise to get the matrix $\boldsymbol{X}_{\boldsymbol{p,BW}} \in \mathbb{R}^{1\times JK_{p,ref}}$.

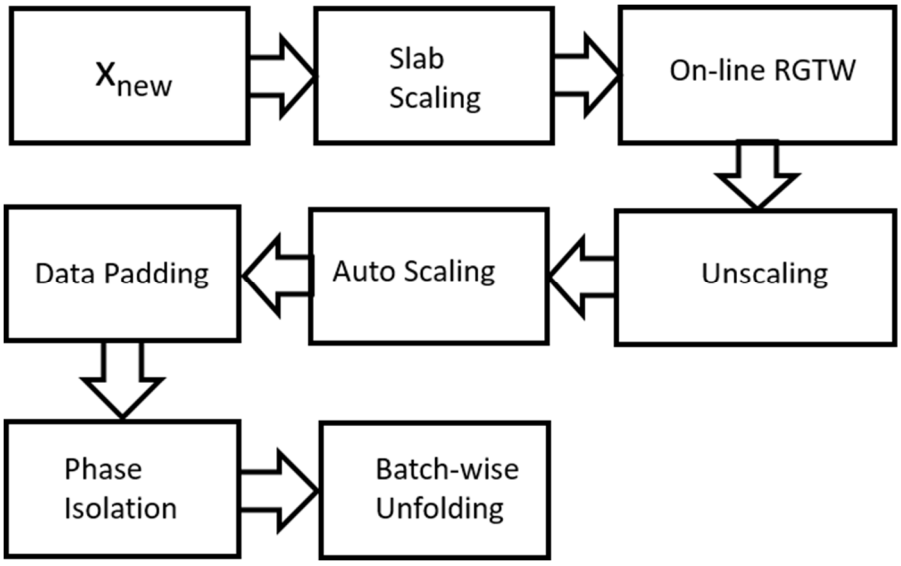


Figure 3.5: On-line Pre-processing Procedure

Model Application

Once the data of the current phase had been identified and isolated, the model could be applied. The following procedure outlines how the models were applied to the data. A detailed description of the routine is given in Appendix F.2.3.

1) Predict the end-of-batch quality $\hat{\boldsymbol{y}}_{\boldsymbol{new}} \in \mathbb{R}^{1\times M}$.
2) Calculate the 99% and 95% confidence limits from this predicted quality.
3) Calculate the ${T_A}^2$ statistic.
4) Calculate the $\mathrm{SPE_X}$ statistic.

3.3 Algorithms Used

This section gives an overview of two algorithms used for pre-processing in this thesis: off-line MPPLS and on-line RGTW. A detailed description of the algorithms is given in Appendix F. Other algorithms, such as off-line Dynamic Time Warping (DTW), had previously been implemented into the Batch Process Monitoring Toolbox (Yzelle, 2013; Barnard, 2015) and was used in this thesis with Sakoe-Chiba local constraint and no constraint on slope. A description of the DTW algorithm is given in Appendix F.1.3.

3.3.1 RGTW Algorithm

The Relaxed-Greedy Time Warping approach used by González-Martínez et al. (2011) was used to synchronise data in an on-line manner. Training data were slab scaled and synchronised off-line using DTW with Sakoe-Chiba local continuity constraint and no constraint on slope (figure 2.7 b). From these synchronised trajectories, the upper and lower limits were determined and used as global constraints as per equations 2.38 and 2.39.

The reference trajectory calculated and used in the off-line DTW was slab scaled off-line in a phase-wise manner, therefore it needed to be unscaled, concatenated and slab scaled globally before it could be used for on-line RGTW. This was due to the fact that the phase information is not always known prior to synchronisation in on-line monitoring and as a result it is not possible to scale the data in a phase-wise manner.

For on-line application, the algorithm processed incoming signal data ($\boldsymbol{X}_{\boldsymbol{new}} \in \mathbb{R}^{k_{ref}\times J}$) as follows:

1) The synchronisation window was created containing the signal path points and all their relevant feasible reference path points defined by the global and continuity constraints from the time interval $(k-\gamma+1)$-th point until the k-th point, where k is the latest signal

time interval and γ is the width of the window, defined by the user. If the window width γ was larger than k then the synchronisation window started at the 1st path point $c(1) = [1,1]$.

2) The optimum path from the starting point of the window to each feasible reference endpoint corresponding to the k-th signal point was found using the cumulated distance measure of equation 2.33. The normalisation factor of equation 2.32 was used where K_n was the number of signal point in the window and K_{ref} was the number of reference points from the starting point of the window to the endpoint.
3) The cumulated distances for each endpoint were compared and the path corresponding to the minimum cumulated distance was chosen as the optimal window path. The final path and warped signal $\boldsymbol{X}_{\gamma,sync}$ was then updated from the reference path point of the $(k - \gamma + 1)$-th point to k_{ref}, the corresponding reference trajectory of the k-th point.
4) The optimal endpoint was compared to the global constraints at the k-th point, and if the point lay on one of the boundaries then the constraints were updated as per equations 2.39 and 2.40.
5) If k was larger than γ then a new starting point for the window was chosen as the optimal path point at the $(k - \gamma + 2)$-th point.

3.3.2 MPPLS Algorithm

The Multi-phase Partial Least Squares algorithm proposed by Camacho et al. (2008) was used to determine the optimal phase divisions for statistical analysis (section 2.5.3). The only unfolding method used for this analysis was batch-wise unfolding, which obviated much of the need for the merging algorithm. A detailed description of the algorithm implementation is given in Appendix F.4.1.

Once the training data had been synchronised off-line, the MPPLS algorithm was used. The parameters k_γ, a_{ini}, T and $minL$ are shown in table 3.1. The initial number of principal components (PCs) to use in the procedure was set to 1 unless otherwise stated. Changing initial number of PCs can change the resulting subdivision, which adds flexibility to the routine. The parameter k_γ weights the choice of adding PCs over partitioning the data by a factor. This favours the addition of PCs over subdivision, and ensures subdivision is done only when the improvement in prediction is greatly enhanced. The choice of improvement threshold T allowed for reasonable but not excessive phase division, and prevents from over-fitting PCs to the training data. The minimum length of the segments also prevents excessive subdivision. The parameters were chosen

differently for the case studies considered due to the lengths of their operational phases. The case studies are introduced in section 3.5.

Table 3.2: Parameters Used for MPPLS Algorithm

Parameter	Value
a_{ini}	1
k_γ	2
T	0.01
$minL$ (Case Study 2)	10
$minL$ (Case Study 2)	50

Following this, the algorithm processed the auto scaled data matrix $\underline{\boldsymbol{X}} \in \mathbb{R}^{I \times J \times K_{ref}}$ as follows:

1) First, the data were unfolded to form the matrix $\boldsymbol{X_{BW}} \in \mathbb{R}^{I \times JK_{ref}}$.
2) A PLS regression model was fitted using a_{ini} latent variables. The Quadratic Prediction Error (QPE) was calculated as the Mean Squared Error (MSE) calculated for each batch using leave-one-out cross validation (LOOCV).
3) A principal component was added to the model and the QPE was calculated as above.
4) The data were split according to the low-level routine in section 2.5.3:
    a) The QRE was chosen as the proportional sum of the of the mean squared error for each segment with respect to the batch length. LOOCV was not used in this step.
    b) The maximum β^k was chosen as the optimal splitting point at time k.
5) The QPE was then calculated at the optimal splitting point and compared to the QPE calculated in step 3 and the parameter T as per the high-level routine.
6) The model was updated appropriately and steps 3-6 were repeated according to the high-level routine. If no model update was required, the model and phase boundaries were returned to be used as the appropriate phase division for end-of-batch quality prediction

3.4 Analysis Procedure and Metrics Used for Analysis

For on-line monitoring, MPLS models trained on NOC data were applied to the test data. NOC data were pre-processed and models were generated using off-line methods. Off-line

synchronisation was done using DTW as per the off-line training of the RGTW approach in section 3.2.2. The following aspects needed to be assessed:

1) Accuracy of on-line synchronisation. This was assessed using the Pearson's Correlation Coefficient (section 3.4.1).
2) Computational speed of on-line synchronisation. This was assessed using the times recorded by the computer's internal clock (section 3.4.4).
3) Prediction accuracy of modelling methods. This was assessed using the mean squared error of the predictions of the product quality (section 3.4.2).
4) Accuracy of fault detection. This was assessed using the different alarms triggered (section 3.4.7).
5) Accuracy of end-of-batch prediction. This wasa assessed using the deviations of the predicted values from their true values (section 3.4.8).

On-line testing was done on each of the model sets and for varying window widths of the RGTW algorithm. The results of the analysis were stored and compared using the metrics below.

3.4.1 Pearson's Correlation Coefficient (PCC)

The PCC is a measure of the degree of linear correlation between two vectors. During synchronisation, a set of path points are obtained by linking an index on the incoming signal trajectory to an index on the reference trajectory, called the synchronisation path. In order to assess the accuracy of the paths obtained using the RGTW algorithm versus the paths obtained using off-line synchronisation, the degree of linear correlation is a reasonable metric to use. This is because paths that are similar usually exhibit the same progression through the synchronisation grid. Therefore, the accuracy of linear correlation between the paths of a trajectory synchronised in two different ways was assessed using the PCC.

The comparison was performed on the training data, therefore the search space limits used in the RGTW algorithm were obtained from the DTW training. This could have influenced the accuracy of the results slightly as all of the synchronisation paths would have fallen within the initial search space and no updating of the limits is necessary to find the optimal path. However, this only applies to extreme cases, and, coupled with the significant amount of time taken to synchronise large amounts of new data off-line, the decision was made just to use the training data.

González-Martínez et al. (2011) used the similarity index (equation 2.40) to assess the accuracy of the synchronisation compared with off-line DTW, which was the product of the PCCs across every batch of training data. This provides a reasonable comparative when considering one set of

training data, but is not normalised. As a result, training data with more batches will have a lower overall similarity index, even if the individual PCCs were similar. Therefore, the metric used to assess the overall linear correlation of the synchronisation methods in this thesis was the geometric mean:

$$mean(PCC) = \left(\prod_{i}^{I} PCC(\boldsymbol{f}_{i,DTW}, \boldsymbol{f}_{i,RGTW})\right)^{1/I} \quad \textbf{3.3}$$

3.4.2 Mean Squared Error (MSE)

During training of the models, the accuracy of the predictions was assessed using the Mean Squared Error in for the quality variable:

$$MSE_y = \frac{\sum_i^I \sum_{k=1}^K (y_i - \hat{y}_{ik})^2}{I\,K} \quad \textbf{3.4}$$

3.4.3 Analysis of Variance (ANOVA) Tests

ANOVA tests were used to assess statistical significance between approaches. The Least Significant Difference (LSD) intervals were calculated if the null hypothesis (p-value > 0.05) was rejected following the ANOVA test. The LSD interval give an indication of which of the groups tested differ from the other groups. The LSD interval between two groups (A and B) was calculated by

$$LSD_{A,B} = t_{0.05/2,DFW}\sqrt{MSW\left(\frac{1}{n_A} + \frac{1}{n_B}\right)} \quad \textbf{3.5}$$

where $t_{0.05/2,DFW}$ is the critical value of the t-distribution at significance $0.05/2$ and the DFW degrees of freedom, DFW is the number of degrees of freedom within the groups (obtained from the ANOVA test), MSW is the mean square within (obtained from the ANOVA test); and n_A and n_B are the number of samples of the groups A and B, respectively. The interval shows the distance from the means of the group A and B where there is no statistical significance. If two groups overlap, then the null hypothesis of no statistical significance cannot be rejected, as shown for groups 1 and 2 in figure 3.6. If the two groups do not intersect, the null hypothesis between the groups can be rejected and statistically significant differences are apparent (as shown for group 3 and the other two groups).

ANOVAs were done to assess the statistical differences between synchronisation paths, as well as the prediction errors obtained for the different models over the length of the batch. When testing

for significance of the synchronisation paths, the Fisher z-transformed PCCs were used. This corrects the PCCs to follow an approximately normal distribution, allowing for the hypothesis test to be performed.

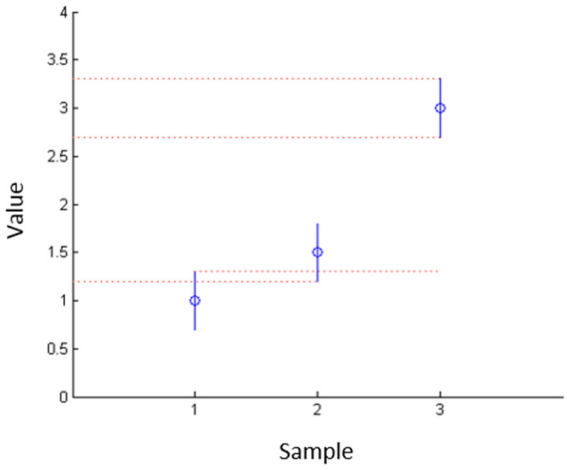


Figure 3.6: LSD Intervals Example showing Statistically Significant Differences for Group 3

3.4.4 Computational Expense

In order to measure the computational expense of the methods, the time taken to process the data every time a new measurement was available was recorded using the computer's internal clock. The details about the hardware and software used are given in Appendix E. Generally, the maximum time taken to produce a result was considered as every result should be available before the next measurement is available to be processed on-line.

3.4.5 Hotelling's T^2 Statistic

The estimated variance s_{t_a} in equation 2.19 was calculated from the training data for every reference time point. For the k-th time point, matrices $\boldsymbol{X_k} \in \mathbb{R}^{I \times kJ}$ and $\boldsymbol{R_k} \in \mathbb{R}^{I \times kJ}$ were used to calculate the scores matrix $\boldsymbol{T_k} \in \mathbb{R}^{I \times A}$ by:

$$\boldsymbol{T_k} = \boldsymbol{X_k}\,\boldsymbol{R_k} \tag{3.6}$$

The variances were calculated and stored. The constant c was also calculated and stored as:

$$c = \frac{I \cdot (I - A)}{A \cdot (I^2 - 1)} \tag{3.7}$$

Once a new measurement had been pre-processed, the Hotelling's T^2 statistic was be generated. The scores $\boldsymbol{t_k} \in \mathbb{R}^{1 \times A}$ from the new data $\boldsymbol{x_{new}} \in \mathbb{R}^{1 \times kJ}$ were calculated by:

$$\boldsymbol{t_k} = \boldsymbol{x_k}\,\boldsymbol{R_k} \tag{3.8}$$

The means calculated from the NOC data were subtracted from the scores vector and a vector the Hotelling's T^2 statistics for every retained component was determined by equation 2.19, where t_a is the value in the a-th column of the vector $\boldsymbol{t_k}$ and ${s_{t_a}}^2$ is its corresponding variance. This statistic was then multiplied by the constant c.

3.4.6 Squared Prediction Error (SPE_x) Statistic

The instantaneous SPE_x statistic calculated for the latest available measurement $\boldsymbol{x_{new,k}} \in \mathbb{R}^{1\times J}$ at time k is given by:

$$SPE_x = \sum_{j=1}^{J} \left(\boldsymbol{x_{new,k}} - \hat{\boldsymbol{x}}_{\boldsymbol{new,k}}\right)^2 \qquad \textbf{3.9}$$

The vector $\hat{\boldsymbol{x}}_{\boldsymbol{new,k}}$ of values predicted by the model were calculated first by determining the scores from the loading matrix of the trained model $\boldsymbol{P_k}$, Regression Weight Matrix $\boldsymbol{R_k}$ at the k-th time point and $\boldsymbol{x_{new}} \in \mathbb{R}^{1\times kJ}$:

$$\hat{\boldsymbol{t}} = \boldsymbol{x_{new}}\, \boldsymbol{R_k} \qquad \textbf{3.10}$$

The resulting prediction vector $\hat{\boldsymbol{x}}_{\boldsymbol{new}}$ was determined by:

$$\hat{\boldsymbol{x}}_{\boldsymbol{new}} = \hat{\boldsymbol{t}}\, {\boldsymbol{P_k}}^{\boldsymbol{T}} \qquad \textbf{3.11}$$

The columns specific to the latest time point, from *(k-1) * J + 1*-th to the *kJ*-th column, of $\boldsymbol{x_{new}}$ and $\hat{\boldsymbol{x}}_{\boldsymbol{new}}$ were used to calculate the SPE_x in equation 3.4.

3.4.7 Fault Detection

The Hotelling's T^2 and SPE_k statistics were used to assess the ability of the models to detect faults, and to identify any false alarms that could occur. Control Limits were defined for each of the statistics for the 95% confidence intervals using the training data. A problem with using the training data to calculate the control limits is that the model fits the training data particularly well, therefore the limits may be much lower than they would be when applied to other NOC data. The reason this was done was because pre-processing extra data off-line was computationally expensive and the pragmatic decision had to be made only to use the training data. In practice, the limits should be set based on separate NOC validation data to test the actual alarm rate. This mainly affected SPE_x statistic as opposed to the T_A^2 statistic, as the limits for the T_A^2 statistic were set

using the critical value of the F-statistic, which was based on the number of training batches and PCs retained. The limits were calculated using equations 2.22 – 2.25. A detailed explanation of their calculation is given in step 4 of Appendix F.1.2.

A fault was detected if one of the statistics exceeded the control limits. The total number of faults detected and the corresponding time interval was recorded. These faults were compared to the actual number of faults that occurred in the process. A false alarm occurred if an alarm was recorded, but no fault occurred. A missing alarm was said to occur if alarm was recorded after an actual fault had occurred. The time delay between an actual fault occurring and three consecutive detections of the fault was also reported on, called the detection delay.

The false alarm rate (FAR) is also reported on for the NOC test batches. This is defined as the number of false alarms during the batch run divided by the number of intervals in the run. For NOC batches, this number should be close to the imposed significance level of the control limits. If the number is too high, the limits should be adjusted. For the purposes of this thesis, the FAR was reported on and suggestions were made if this number was significantly higher than the imposed significance level of the control limits.

For faulty test batches, the number of false alarms during the period of normal operation was also reported on. Using the FAR in this case is inappropriate as no false alarms could be reported after the fault occurred. Using only the interval before the fault occurred to determine the FAR would skew the FAR to certain parts of the process. Therefore, the absolute number of faults was discussed in context of the case study.

The missing alarm rate (MAR) can also be defined as the number of missing alarms divided by the number of intervals where the fault occurred. These are discussed in context with the total number of missing alarms.

Assessing the types of alarms and alarm rates helps assess the performance of the following aspects of the on-line monitoring platform:

- Whether the models are accurate representations of the NOC data
- Whether on-line synchronisation inaccuracies affect the number of alarms detected
- Whether faults can be detected with reasonable certainty

3.4.8 Deviation of Predictions from their True Values

When assessing the accuracy of the end-of-batch predictions, the deviations of the predictions from the true value of the product variable were calculated for every time interval. These values

were plotted to illustrate the deviation along time. The squared differences were not required in this instance because the deviations were assessed for each individual NOC test batch, and the direction of the deviation provides insight into how the predictions evolve over time. This is important because as more information becomes available in the modelled phase, fewer zeroes have to be imputed. The trends of the predictions over the course of a modelled phase can offer insight into the nature of the model.

3.5 Case Studies Used

Three studies were considered in this thesis:

1) The first case study was a simulation of a three-tank system, which was a simple, non-linear system containing three tanks that are filled with a fluid. There are five measured variables in the system, and one quality variable. The small number of variables makes the system relatively easy to study
2) The second case study was a simulation of the penicillin cultivation process. More measured variables (11) and quality variables (5) were considered, making this case study more complex than the former. It is also a well-studied process in batch process monitoring, making it ideal to be used in this study
3) The third case study was actual data from the final concentrate dissolution process. This case study was used to investigate the effect of the techniques on actual plant data. The process consisted of 5 process variables and 3 quality variables. On analysis of the data, it was discovered that the quality variable could not accurately be predicted from the measured variables (shown in Appendix D), therefore no fault detection and end-of-batch quality was performed.

CHAPTER 4 Case Study 1 – Three Tank Simulation

The three-tank system (figure 4.1) is a multivariate, nonlinear system used as a benchmark to evaluate process modelling and monitoring methods (Chalupa, Novák, & Bobál, 2011; Kanagasabai & Jaya, 2014; Sobhani, 2011). While this process is a continuous process, Lu et al. (2004) used this system as a verification of the sub-PCA modelling approach (a batch process monitoring approach), where the tanks levels were brought from their initial conditions to certain set points to simulate a transient process. A simulation of this system was developed for this project to evaluate the on-line synchronisation, end-of-batch prediction and fault identification performance.

4.1 Process Description

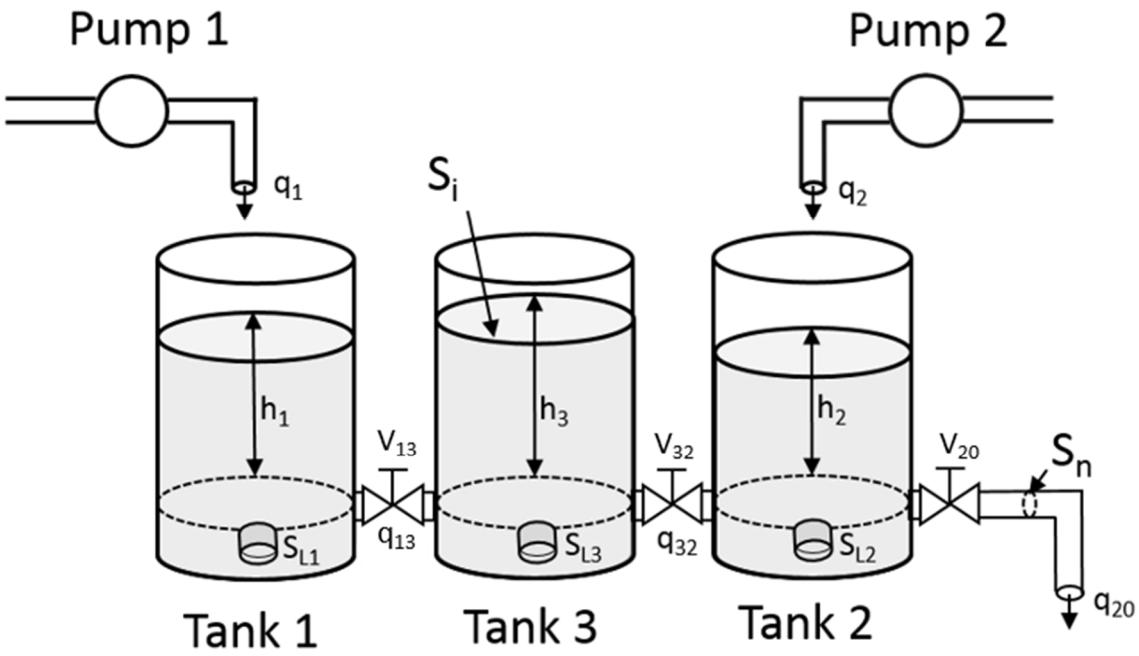


Figure 4.1: Schematic of Three-Tank System

The process consists of three tanks 1, 3 and 2; labelled from left to right. The tanks are connected by valves. Fluid entered the system from the two pumps and exited the system through the outlet valve, V_{20}. Holes with variable cross-sectional area were included in each tank at the same level as the pipes to simulate leaks in the system $(S_{L_1}, S_{L_2}, S_{L_3})$.

Mass balances around the system volumes yielded the following ordinary differential equations:

$$S_1 \frac{dh_1}{dt} = q_1 - q_{13} - q_{L_1} \qquad 4.1$$

$$S_3 \frac{dh_3}{dt} = q_{13} + q_{32} - q_{L_2} \tag{4.2}$$

$$S_2 \frac{dh_2}{dt} = q_2 - q_{32} - q_{20} - q_{L_2} \tag{4.3}$$

q_1 and q_2 are flow rates from the pumps; q_{13}, q_{32} and q_{20} are the flows between the tanks, which are dependent on the differences of heights between the tanks; and q_{L_1}, q_{L_2} and q_{L_3} are the flow rates from the potential leaks within the system. S_i is the cross-sectional area of the i-th tank. The cross-sectional areas of the tanks are the same ($S_1 = S_2 = S_3 = S_T$), thus the flow between tanks can be represented for valve V_n as follows

$$\frac{q_n}{S_T} = k_n \sqrt{|\Delta h|}\, \frac{\Delta h}{|\Delta h|}, \qquad n = 13,32,20 \tag{4.4}$$

where

$$k_n = \mu_n \frac{S_n \sqrt{2g}}{S_T}, \qquad i = 1,2,3 \tag{4.5}$$

Here, S_n is the cross sectional area of the opening between the two tanks, μ_n is a valve flow coefficient and g is the gravitational constant. $S_n$and S_T are physical parameters of the process and were treated as constants (table 4.1). Δh refers to the difference in levels between the tanks on either side of the valve. The direction of flow was evaluated by dividing the change in level by its absolute value $\left(\frac{\Delta h}{|\Delta h|}\right)$. The flow from leaks was given by

$$q_{L_i} = \frac{S_{L_i} \sqrt{2g}}{S_T} \sqrt{h_i}\,, \qquad i = 1,2,3 \tag{4.6}$$

where $S_{L_i}$is the cross-sectional area of the hole introduced through the fault valve of tank *i*. The area of the hole could be changed on a run-to-run basis to simulate leaks of different magnitude. For NOC data where no leaks occur, this parameter was set to zero.

Table 4.1: Constants used in Three-Tank Simulation

Parameters	Description	Value(s)
$S_1 = S_2 = S_3 = S_T$	Tank cross-sectional area	$0.0177\ m^2$
$S_n = S_{13} = S_{32} = S_{20}$	Inter-tank cross-sectional area	$7.854 \times 10^{-5}\ m^2$
$\mu_{12} = \mu_{20}$	Inter-tank valve coefficients	1
μ_{32}	Inter-tank valve coefficients	0.5

A batch run consisted of raising the levels of h_1 and h_2 from $h_1 = h_2 = h_3 = 0$ to their predefined set points by pumping fluid from the pumps. Pumps were manually operated, both starting at 100% and ramping down to 30% flow within a specified interval. Closed-loop control was implemented on the levels of h_1 and h_2 by adjusting the valve flow coefficients μ_{13} and μ_{20}, respectively, by a linear factor as the flow through the valves was assumed to be a linear function of the valve position. Variability and limits were introduced to the valves and the pumps to simulate real world dynamics (table 4.2). Random noise was added to the pumps such that the output signal varied within 90-110% of the input signal value.

Table 4.2: Variability and Limits used in Three-Tank Simulation

Component	Type	Value(s)
Pumps	Maximum flow rate	$1.5 \times 10^{-4}\ m^3/s$
	Pump Variability	$90 - 110\%\ of\ input\ signal$
	Pump 1 Flow Ramp Decrease Time	$60\ \pm 5\ s$
	Pump 2 Flow Ramp Decrease Time	$30\ \pm 5\ s$
Valves	Signal Delay Time	$2\ s$
	Valve 13 position range	$35 - 60\ \%$
	Valve 20 position range	$30 - 70\ \%$
	Valve position rate change limit	$5\ \%\ per\ s$
	Set point 1	$0.5\ m$
	Set point 2	$0.4\ m$

4.2 Data Collected During the Process

During the process, five process variables were measured:

1) Level of tank 1
2) Level of tank 2
3) Level of tank 3
4) Pump 1 flow rate
5) Pump2 flow rate

The batch run was completed when the levels were within 0.01 m of their set points and the absolute change in levels with respect to time was less than 0.0001 m/s. The time taken (in seconds) for the batch to run to completion was recorded as the only quality variable for the process. Time was not included as a variable because the final time taken was already included as a quality variable.

Four user-defined phases were identified (figure 4.2):

- When both pumps were running at 100 %
- When pump 1 was running at 100 % and pump 2 was ramping down to 30 %
- When both pumps were ramping down to 30 %
- When both pumps were running at 30 %

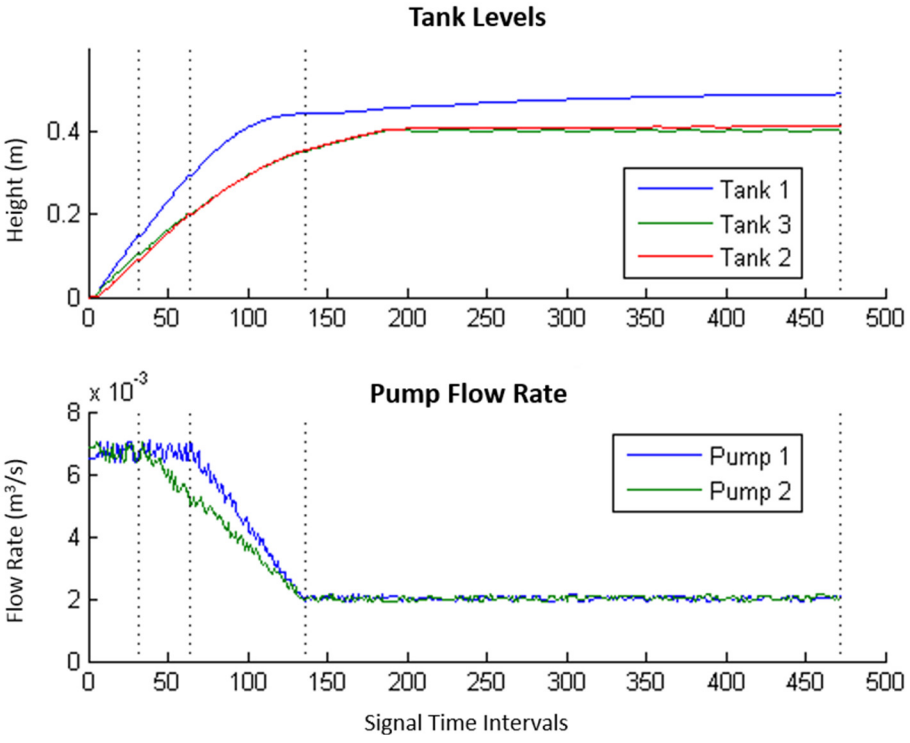


Figure 4.2: Phase Boundaries – Case Study 1

A total of 21 runs were simulated under NOC and used to train the models, similar to the number used by Lu et al. (2004). Two additional runs were simulated as test data: one running at NOC and one running at faulty conditions where a leak occurred ($S_{L_1} = 2.83 \times 10^{-7}\ m^2$) at the 160th time interval (figure 4.3). The fault caused tank 1 to take much longer to reach its set point compared to the other simulations operating under NOC. This fault was chosen because of its effect on the end-of-batch quality variable.

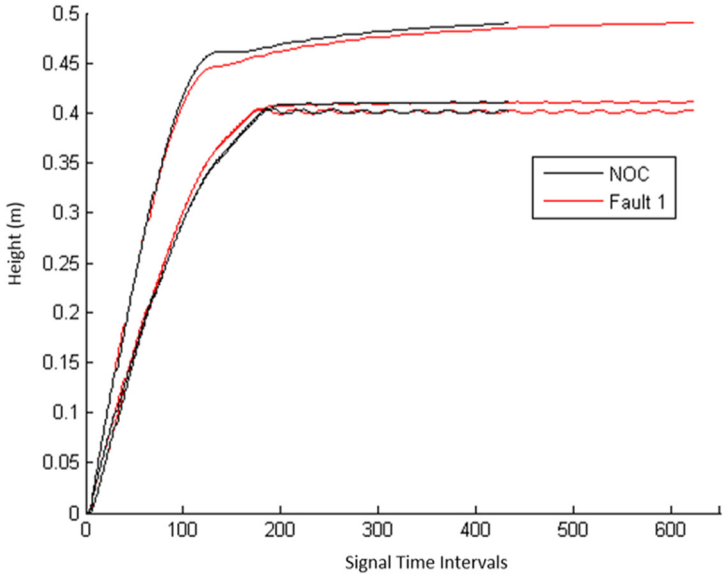


Figure 4.3: Test Batches – Case Study 1

4.3 Batch Trajectory Synchronisation

Due to the variability in the data, the batch trajectories first had to be synchronised to a common reference trajectory. The data used for training were synchronised off-line using dynamic time warping (DTW), and models were built on the synchronised data. During on-line monitoring, it is important that the data be synchronised similarly so that the model is applied to data at a similar point in the trajectory. It therefore makes sense to compare the off-line synchronised data to the same data synchronised in an on-line fashion. The linear correlation of the off-line and on-line warped paths was compared using the Pearson's Correlation Coefficient (PCC) as a metric, and the geometric means of these PCCs over all the batches can be compared for different on-line synchronisation procedures.

The process took approximately 300s to complete, therefore producing a result within 1s was considered adequate. The RGTW approach would be able to produce timely results at the expense of warping accuracy, therefore a compromise needs to be made. Figure 4.4 shows the paths of the off-line synchronised data using DTW.

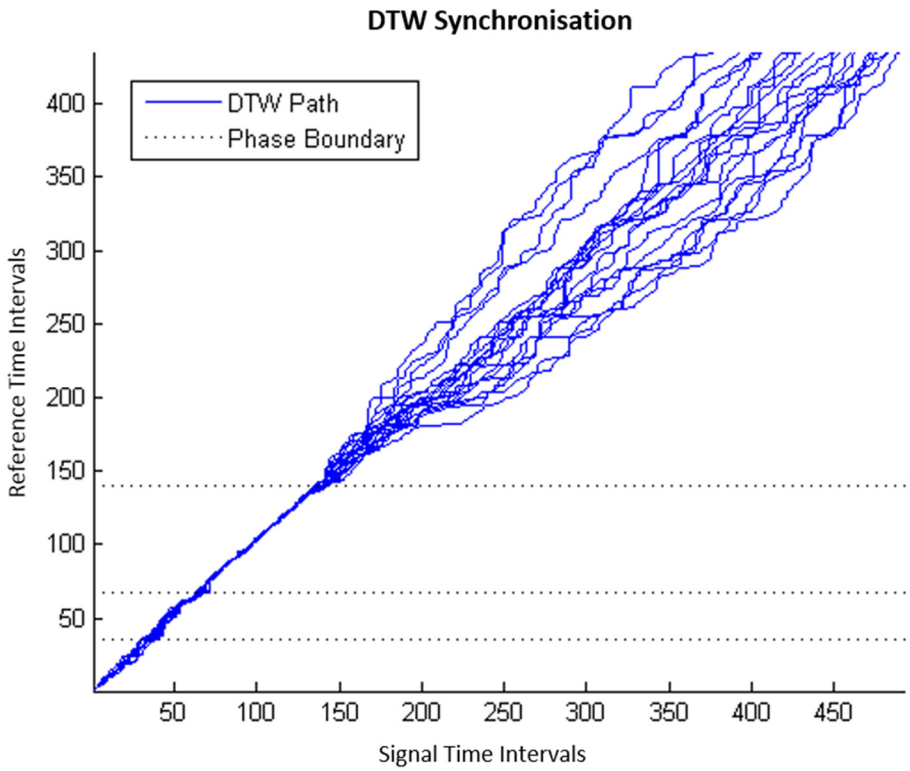


Figure 4.4: Off-line Synchronisation Paths – Case Study 1

In the first two phases, slight warping occurs to stretch and compress the trajectories due to the differing phase durations (caused by the variability around the pump 1 decrease time). Very little warping needed to be done in the third phase, except near the beginning and end. The most significant warping occurred in the fourth phase. This is probably because the tanks levels were left to reach their set points while the pumps were running at constant rates. The final levels were reached at different times across the batches, so the trajectories had to be compressed or expanded such as to be aligned with the reference batch. Additionally, the Euclidean distance used to compare batch trajectories in the DTW algorithm was reported by Gins et. al (2012) potentially to cause pathological warping in when the trajectories are flatter and proposed Hybrid Derivative Time Warping (HDDTW) in an attempt to increase the accuracy during these parts of the process. This is a potential issue that may have affected the warping in these regions.

On-line synchronisation using the RGTW approach for window size of $\gamma = 3, 5, 10, 20, 50$ as well as setting the window width to be larger than the largest trajectory $(\gamma = \infty)$, which is a similar approach to on-line DTW, but using the search space limits of the RGTW approach. This on-line approach was applied to the full trajectory as opposed to at each interval as the final path was of concern, and the computational time taken to synchronise the trajectories in this was significant. Differences in warping are apparent when comparing the paths for $\gamma = 5$ and $\gamma = \infty$ (figures 4.5

and 4.6), especially in the fourth phase. For the smaller window sizes, the compression and expansion of the trajectories were not as drastic as off-line synchronisation, indicating that the trajectory paths were optimised locally. Performing the synchronisation over a large window is advantageous here because the dynamics of the process become slower in this region. Furthermore, the sinusoidal variation of the level of tank 3 (which was not controlled) may also have affected the warping for smaller window sizes. The change in gradient of this variable as it varies about a relatively constant value may influence the cumulated distance measure for these small window sizes, considering the other variable trajectories are much more constant. Improper synchronisation in this region would have a negative impact on the accuracy of end-of-batch predictions and fault detection as the model was trained on the off-line synchronised data.

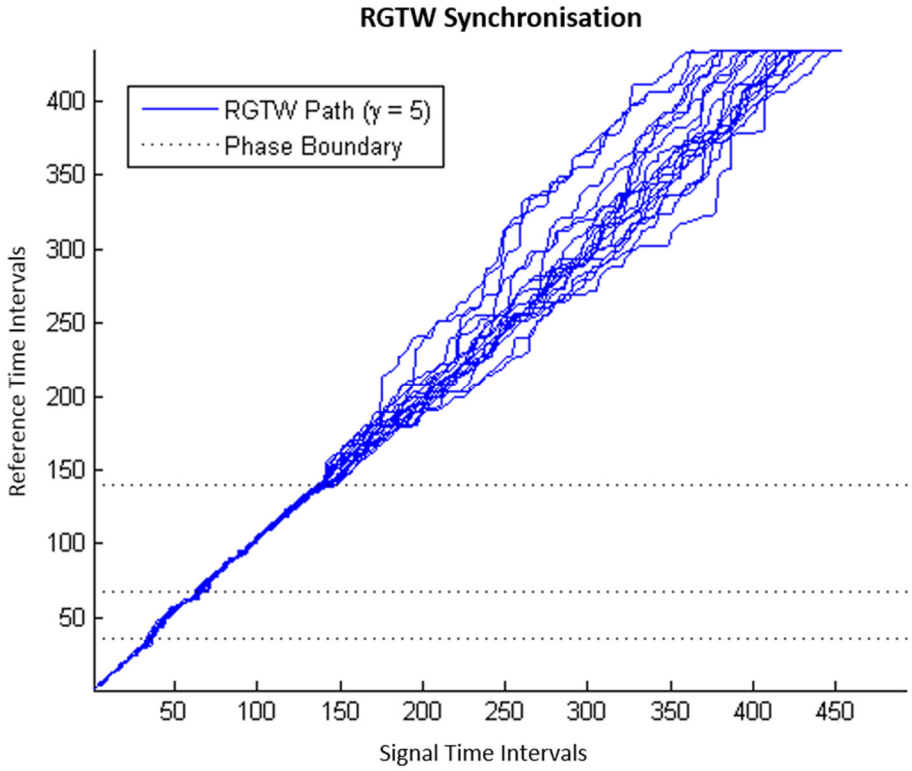


Figure 4.5: On-line Synchronisation paths using RGTW ($\gamma = 5$) - Case Study 1

It should also be noted that even though the off-line DTW algorithm was performed per phase, whereas the on-line approaches were performed on the trajectory as a whole, both approaches showed similar warping patterns across the batch lengths. Off-line DTW was implemented in a phase-wise manner to improve the computational efficiency of the pre-processing, however, it requires known endpoint phase boundaries. The on-line approaches performed similarly in the first three phases without the requirement of these phase boundaries, which is advantageous for on-line implementation.

An added advantage of applying off-line DTW in a phase-wise manner is that slab scaling could be done per phase, which could have improved warping accuracy in the final phase, when the process dynamics were significantly slower than before. The scaling would reflect the dynamics of the phase rather than the dynamics over the entire process.

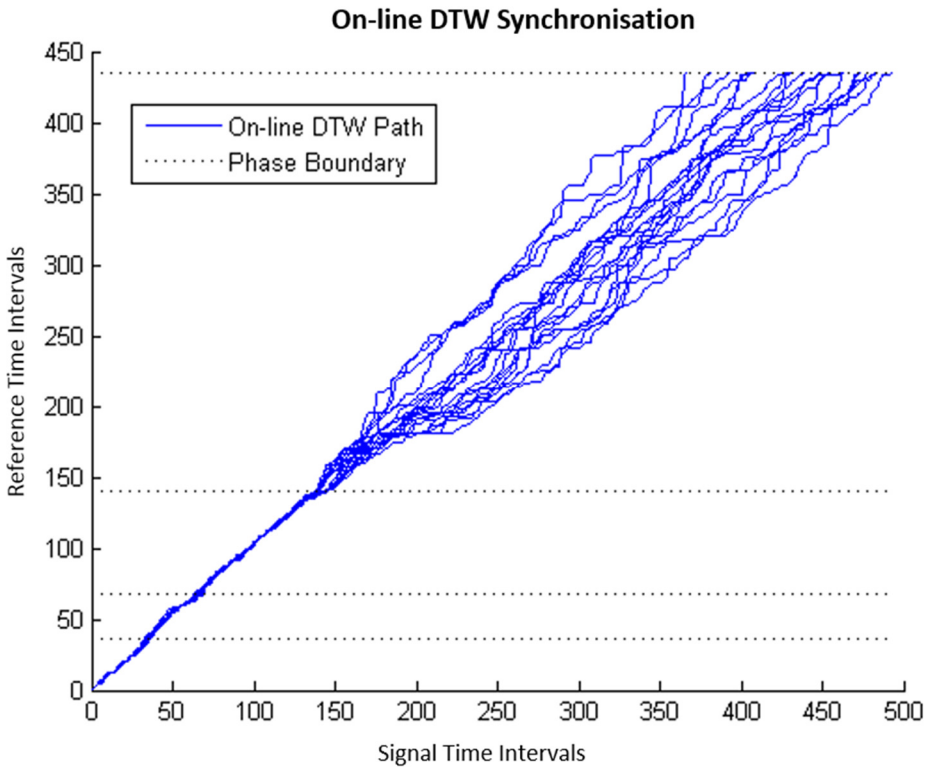


Figure 4.6: On-line Synchronisation paths using on-line DTW ($\gamma = \infty$) - Case Study 1

Table 4.3 shows the geometric means of the PCCs for each window size, compared with off-line synchronisation. Overall, the approaches showed pretty strong correlation, which improved as the size of the window was increased. An analysis of variance (ANOVA) was done on the Fisher z-transformed PCCs to determine statistical significance for p-value < 0.05. The least significant differences (LSD) intervals (figure 4.7) showed statistically significant differences between the two largest $(\gamma = 50, \infty)$ and three smallest $(\gamma = 3, 5, 10)$ window sizes, indicating that the accuracy of warping is much improved for a larger window size.

Table 4.3: Mean PCCs between off-line DTW and on-line approaches – Case Study 1

Synchronisation Method	Geometric Mean	No. of Batches
RGTW (γ=3)	0.9972	21
RGTW (γ=5)	0.9973	21
RGTW (γ=10)	0.9973	21
RGTW (γ=20)	0.9981	21
RGTW (γ=50)	0.9983	21
On-line DTW (γ= ∞)	0.9986	21

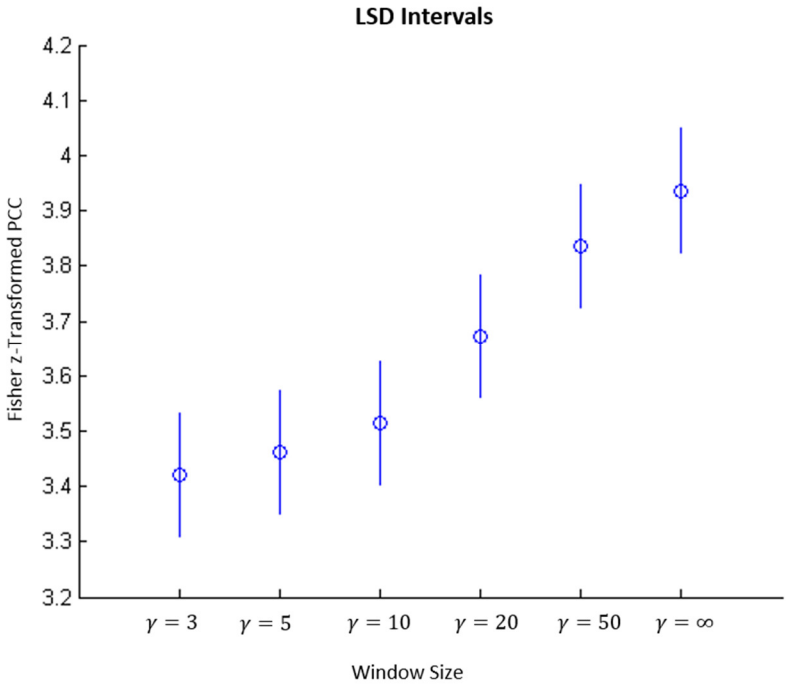


Figure 4.7: LSD Intervals for different window sizes

4.3.1 Computational Expense

Thus far the larger window sizes have shown statistically significant differences in accuracy compared to the smaller window sizes. Larger window sizes increase the search space and potentially the computational time for the trajectories to be synchronised at time interval. A summary of the computational times taken per sample are given in table 4.4.

Table 4.4: Computational Efficiency of On-line Synchronisation – Case Study 1

	$\gamma = 3$	$\gamma = 5$	$\gamma = 10$	$\gamma = 20$	$\gamma = 50$	$\gamma = \infty$
Number of samples	9246	9246	9246	9246	9246	21
Average (seconds)	0.0284	0.03047	0.0563	0.1252	0.5093	20.94
Standard Deviation	0.0104	0.0166	0.0039	0.0978	0.5227	3.056
Minimum	0.0022	0.0022	0.0022	0.0022	0.0028	14.86
Maximum	0.3005	0.0961	0.3972	0.4730	1.7727	25.62

The maximum time taken for the RGTW approach with window widths of 3, 5, 10 and 20 were all under 1 second, which is acceptable for this simulation. Although the mean time taken per sample using a window width of 50 was under 1 second, the maximum time taken was well over that value. In order for a timely result to be available, the computational time for every sample should be under the threshold value for the process.

Using the information gathered above, a conclusion can be drawn that the most appropriate on-line synchronisation approach for this simulation would be RGTW with a window width of 20, as it produces the most accurate results in time for the new measurement to be available. For comparative purposes, on-line monitoring and end-of-batch prediction was done using window widths of 5 and 20 in the following sections.

The next section presents the results of the different models trained by partitioning the data in three ways: using no partitions (i.e. a global model), partitioning according to the operational phases (a phase-wise model) and partitioning according to the correlation phases using the multiphase (MPPLS) algorithm.

4.4 Model Training and Phase Division

MPLS models were trained on the NOC data in order to relate the five measured variables to the quality variable. The effect of building separate models for different phases was also investigated. In order to choose the number of principal components (PCs) to retain for the global and operational phase model, a cross validation approach was used on the NOC training data. A compromise was made between the cumulative variance explained in $\boldsymbol{Y}$ and the overall mean squared error (MSE) between the actual quality variable and the predicted quality. Information about the model sets for each type of partitioning is given in table 4.5.

Table 4.5: Data Partitioning and Cumulative Variance Explained for Different Model Sets – Case Study 1

Phase Division	Phase	Reference Intervals	No. PCs Retained	Cumulative Variance Explained in Y	Cumulative Variance Explained in X
Global	1	1 – 435	2	98.7 %	27.5 %
Operational Phases	1	1 – 36	4	78.2 %	52.2 %
	2	37– 68	3	86.5 %	50.9 %
	3	69 – 140	1	88.8 %	50.5 %
	4	141 – 435	1	94.7 %	12.2 %
MP algorithm	1	1 – 67	1	38.1 %	24.5 %
	2	68 – 435	2	99.2 %	25.8 %

The MPPLS algorithm was initialised with a_{ini} = 1 PC, and was able to find a split that would minimise the Mean Squared Error (MSE) across the entire batch duration. Two correlation phases, split at the 67th interval. This coincides almost exactly with the start of the 3rd phase. At this point, both pumps flow rates have started to decrease and the tank level control is implemented as the levels approach their set points. The tank levels are not only influenced by the pump flow rates, but also by the valve positions between the tanks (which are controlled). This changes the correlation structure of the variables, thus it is an adequate phase division.

The amount of product variance explained in the first correlation phase is fairly low (38.1%), but analysis of the MSEs calculated through cross validation (labelled "test" in the figure 4.8) shows that the lowest value is found when one PC was included in the model. Therefore, the algorithm would not have added more PCs to the model. However, more than 80% of the variance could have been explained if 3 or more PCs were retained (figure 4.9). Choosing a model with 3 or more PCs would probably be more appropriate for this phase, as the slight MSE increase could be justified by the extra variance explained. Starting the algorithm with a_{ini} = 3 PCs returned a global model with no extra PCs added. This is a caveat of the MPPLS algorithm, as sometimes a compromise between variance explained and model accuracy must be made. Incorporating the amount of variance explained into the algorithm may be a way to improve the model performance, although practically this is a difficult task as hard limits or thresholds are difficult to define for such a case.

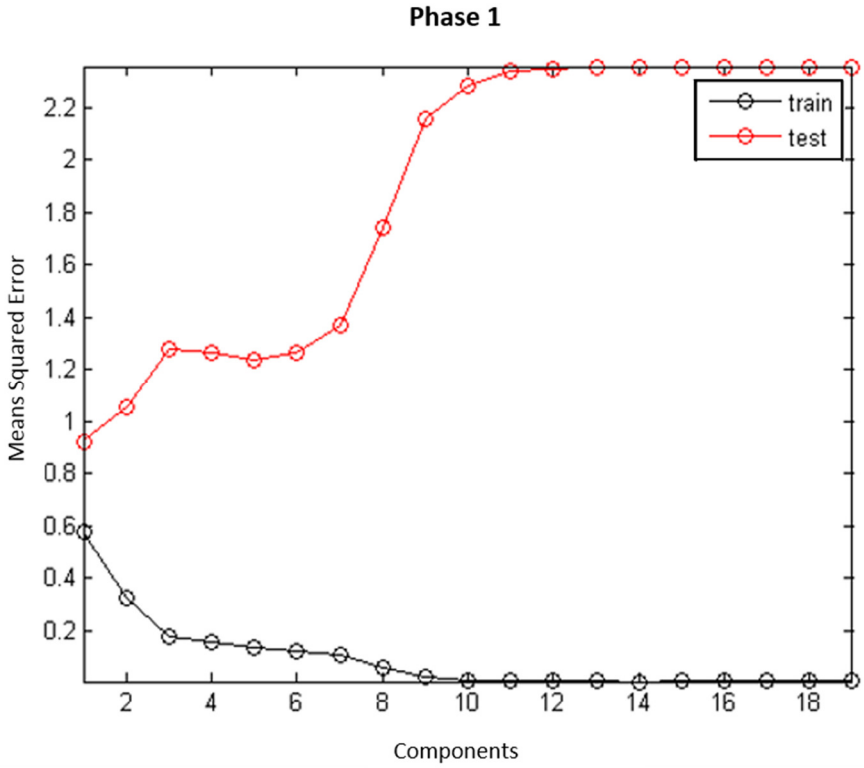


Figure 4.8: MP Algorithm MSEs – First Correlation Phase

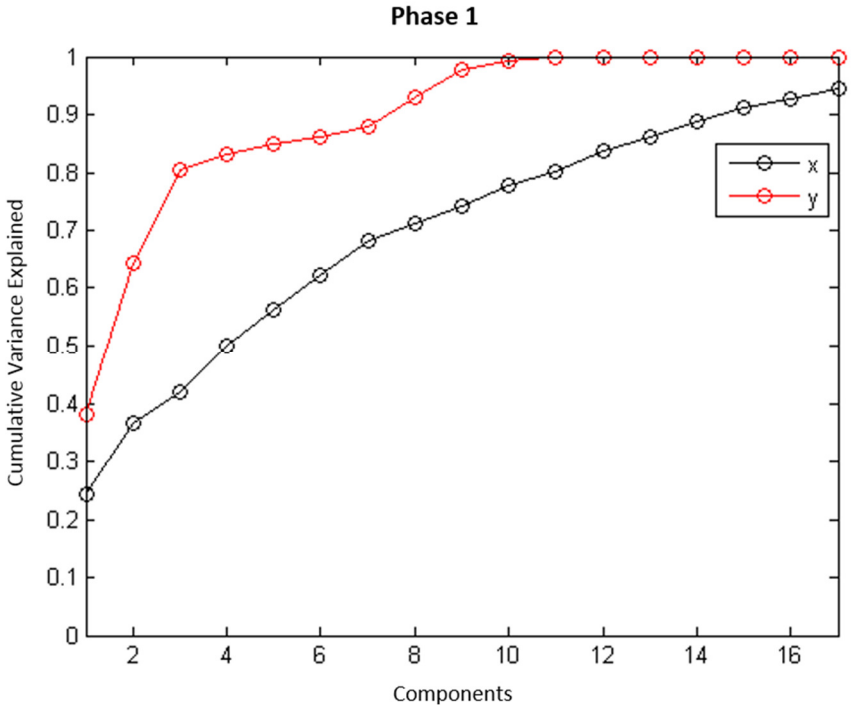


Figure 4.9: Cumulative Variance Explained for First Correlation Phase of MP Algorithm – Case Study 1

The trend of the MSE in the second correlation phase (figure 4.10) showed a decline in the MSE as more components were retained. The improvement of retaining more than 2 PCs was not enough to overcome the specified improvement threshold (T = 0.01, table 3.2), which was appropriate considering the model was able to account for almost all of the variance in the product

variable. The better fit of the model in this phase provided a compromise for the poor fit to the model in the first phase, considering this phase was much longer than the first and predictions at all time intervals were weighted evenly. Weighting the intervals would allow the algorithm to pay more attention to critical-to-quality parts of the process. This may need to be done through process analysis, as was done by Lu and Gao (2005) when defining critical-to-quality operational stages in their stage-based PLS modelling approach.

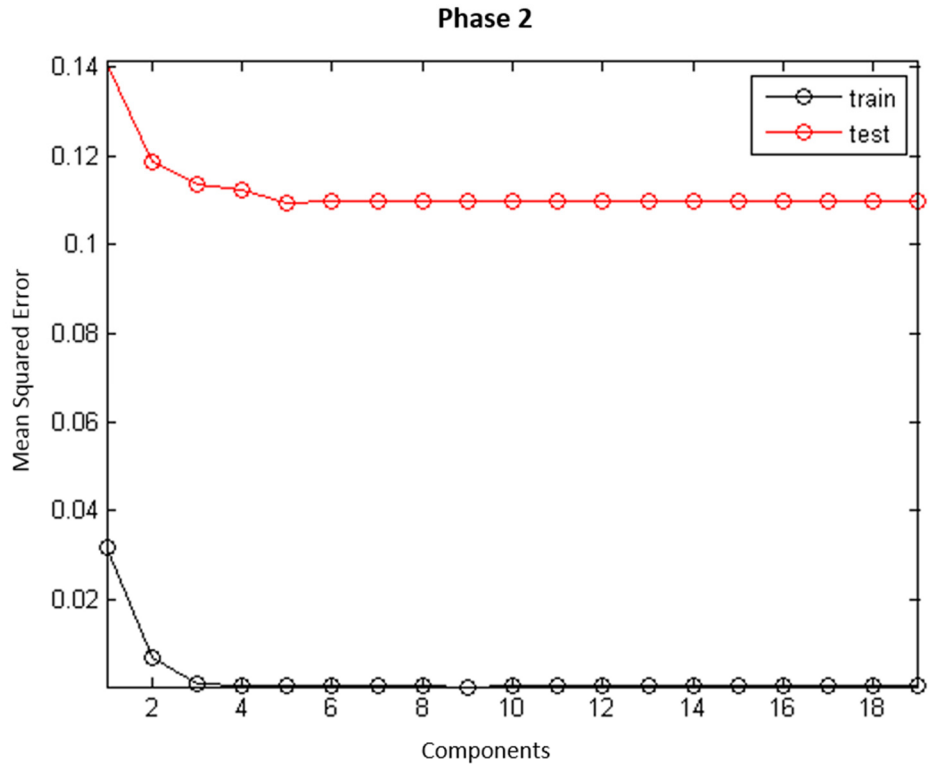


Figure 4.10: MP Algorithm MSEs – Second Correlation Phase

A comparison between the MSEs of the training data for the three approaches with respect to time is presented in figure 4.11 along with the operational phase boundaries. The models generally showed the highest MSE outcomes at the beginning of each phase as most of the data comprised of zeros imputed as missing data.

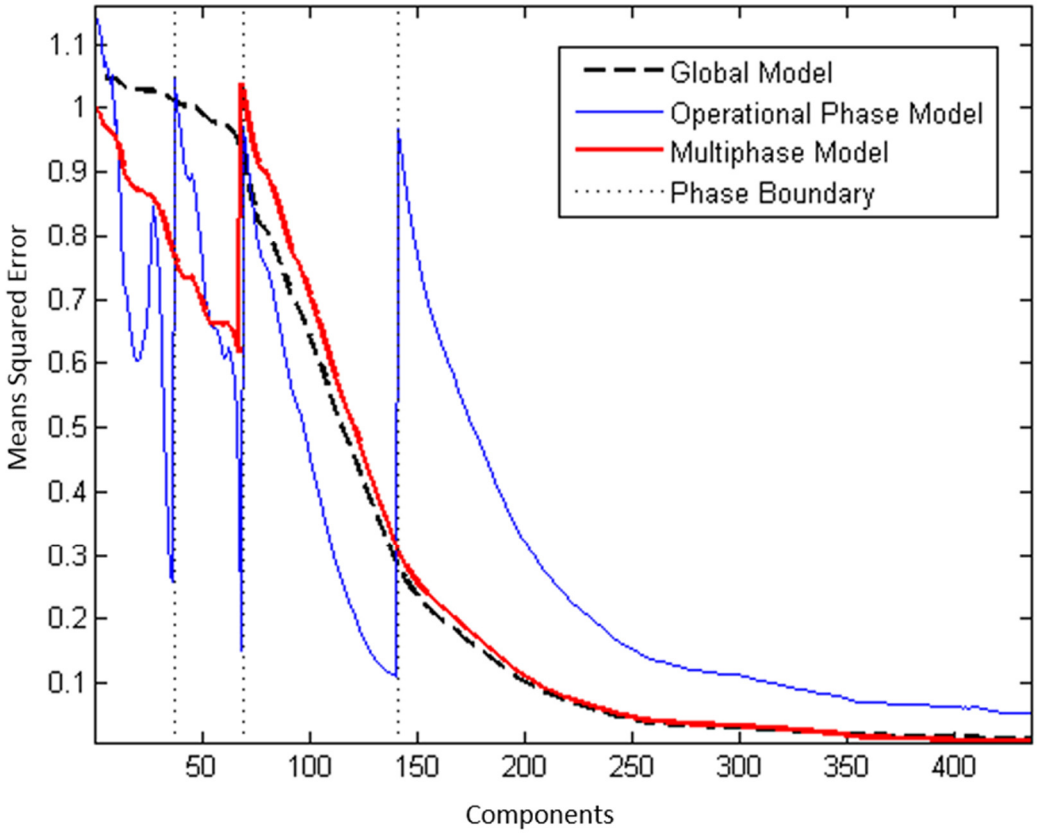


Figure 4.11: MSE Outcomes with respect to Time – Case Study 1

The MSE outcomes of the global and MPPLS models are similar in the second correlation phase identified by the MPPLS algorithm. However, the MPPLS algorithm was able to partition the data and reduce the MSE in the first correlation phase. An ANOVA on the MSEs produced showed no statistically significant differences (p-value < 0.05) among the approaches. Improvements mentioned previously may be needed to enhance accuracy of the predictions.

The following section details the application of these models to the test cases for on-line monitoring and end-of-batch prediction. Although the amount of variance explained in the first correlation phase could be improved by retaining more PCs, this would nullify the automatic nature of the algorithm. In practice, post analysis could be done on the MSEs to improve prediction power of the model(s). Changing the initial parameters of the MPPLS algorithm returned a global model, which was not necessary to include as a global model had already been trained.

4.5 Model Application for On-line Fault Detection and End-of-Batch Prediction

The models trained for the three different cases were applied to the two test cases in an on-line fashion. The SPE_x and Hotelling's T^2 statistics were used to monitor the batches and the alarm rates were determined based on the 95% confidence limits proposed by Nomikos and MacGregor (1995). If the value of either of the statistics exceeded the imposed confidence limits, an alarm was recorded. The predicted end-of-batch quality was reported on for the NOC test batch only, as the model becomes invalid if a fault occurs. On-line synchronisation was done using RGTW with window widths of 5 and 20 intervals to evaluate the effect of on-line synchronisation on the monitoring results. The monitoring and prediction charts for all of these conditions are given in Appendix B.

4.5.1 On-line Fault Detection

The monitoring results for the NOC test data synchronised with window sizes of 5 and 20 are presented below in table 4.6. Every alarm recorded during the NOC test batch was assigned as a false alarm. Every alarm recorded during the fault test batch before the 160th interval (when the fault occurred) was assigned as a false alarm, and any alarm not recorded after the 160th interval was assigned as a missing alarm. The missing alarm rate was calculated using the number of intervals after the fault occurrence (462).

Table 4.6: Fault Detection Results for NOC Test Batch – Case Study 1

Window Size	No. Intervals	Phase Division Method	Number False Alarms	False Alarm Rate
5	433	Global	154	0.356
		Operational Phases	130	0.300
		MP Algorithm	133	0.307
20	433	Global	143	0.330
		Operational Phases	32	0.073
		MP Algorithm	33	0.076

A high false alarm rate (FAR) was recorded for the NOC test batch when synchronising using a window width of 5. The majority of these false alarms occurred through the SPE_x statistic at the end of the batch, as shown in figure 4.12 for the operational phase model. The reason for these false alarms is the suboptimal synchronisation at the end of the batch run. Figure 4.13 shows fewer false alarms in this region for a window width of 20. A plot of the synchronisation paths the two window sizes (figure 4.14) shows that different paths were taken at the 296th signal interval. The greedier approach of the smaller window size assumes the batch to be further along than it actually is, and excessively expands the trajectory. This is probably due to the slower dynamics of the variables in this part of the process, which the smaller window size cannot correct for. This suboptimal warping may need to be taken into account when defining the monitoring chart limits. The limits were calculated based on data synchronised off-line, but if the greedy approach is used on the calibration data, the suboptimal warping may be taken into account. However, this could leave the model susceptible to missing alarms.

The reason for the global model not significantly reducing the false alarm rate when a larger window width was used is due to the second operational phase of the process. The limits defined in this phase for the global model are quite low compared to the other models, therefore more false alarms were recorded. This is a symptom of using the training data to define the limits, as opposed to a separate set of calibration data. These alarms should be recalibrated to relax the limits somewhat.

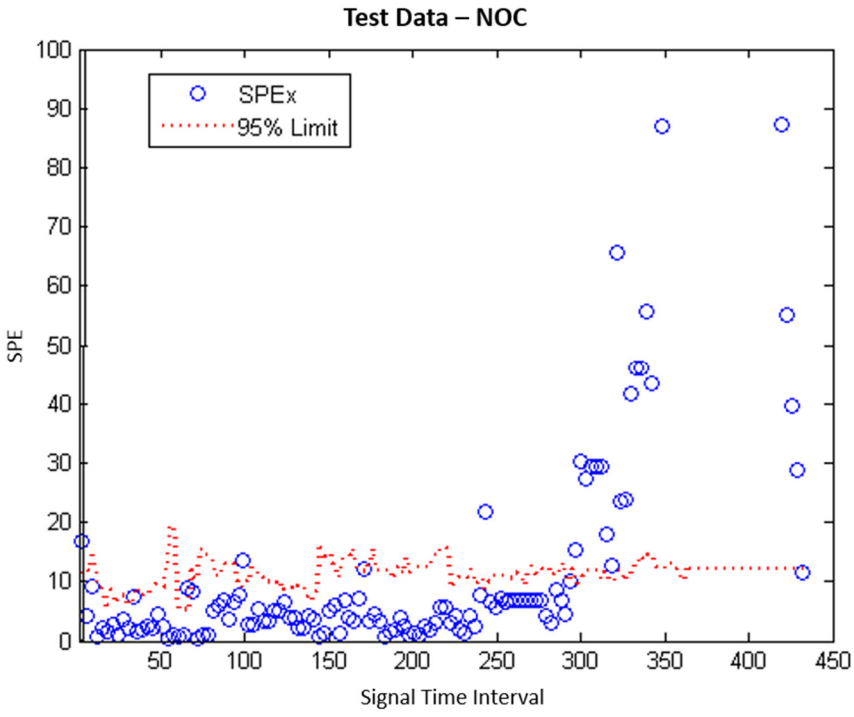


Figure 4.12: SPE Chart for NOC Test Data (Operational Phase Models, γ = 5)

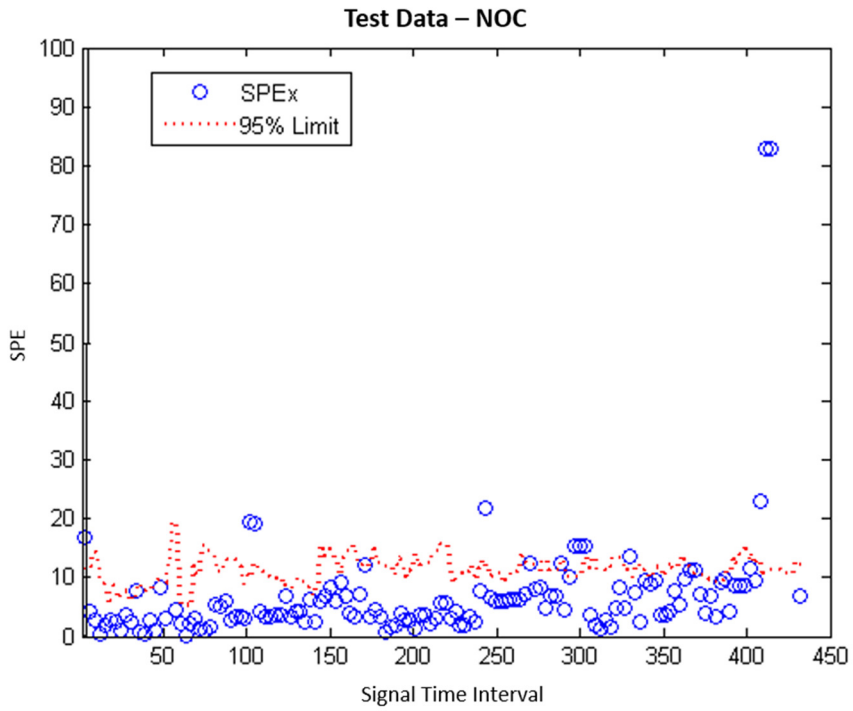


Figure 4.13: SPE Chart for NOC Test Data (Operational Phase Models, γ = 20)

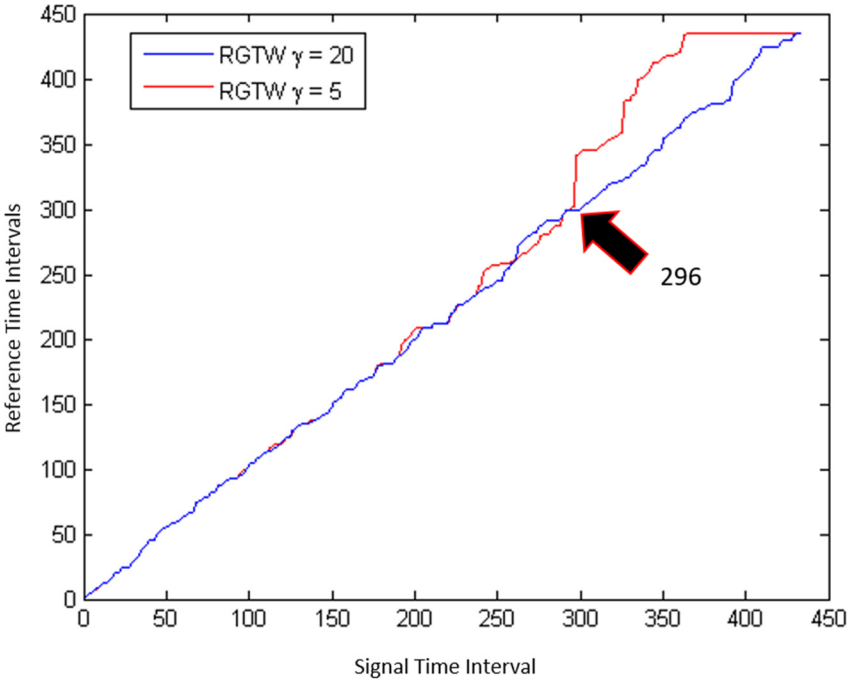


Figure 4.14: Synchronisation Path Deviations for NOC Test Batch

The fault identification results for the faulty batch for both window size is show in table 4.7. The fault introduced was a small leak in tank 1, which resulted in the process taking much longer to reach its completion time. Much better performance is seen in for the smaller window size compared with the NOC batch.

Table 4.7: Fault Detection Results for Fault Batch – Case Study 1

Window Size	No. Intervals	Phase Division Method	False Alarms	Missing Alarms	Detection Delay
5	622	Global	12	76	34
		Operational Phases	2	45	52
		MP Algorithm	20	48	41
20	622	Global	81	172	37
		Operational Phases	8	138	52
		MP Algorithm	20	273	54

The fault was detected 36-54 intervals after they occurred for all the approaches the reason for this delay probably lies with the nature of the fault, as is didn't drastically change the variable trajectories. The fault was identified only after tank 2 had reached its set point, and at this point the deviation of tank 1 from its set point was great enough that the data did not fit the model.

After the fault had been identified, the larger window size allowed the trajectories to be compressed and many missing alarms occurred, as the synchronisation paths show in figure 4.15. The fault that occurred affected the time taken for the batch to reach completion, and synchronisation warps the trajectories to a pseudo time, hence this information was lost when the trajectory was compressed. Initially the fault was detected, but as the process continued, the larger window size allowed the signal trajectory to be compressed, resulting in missing alarms. This did not occur for the smaller window size. The synchronisation paths show that a relatively consistent linear path was maintained at this part of the process. Using a window size of this length might be advantageous at this part of the process, as it can detect faults such as this that affect the time taken for the batch to reach completion. The false alarms that may occur at the end of the process could be obviated by using a dynamic window size that changes from 5 intervals to 20 intervals once a particular reference point has been reached. Alternatively, time could be included as a variable, considering it is such an important part of the process.

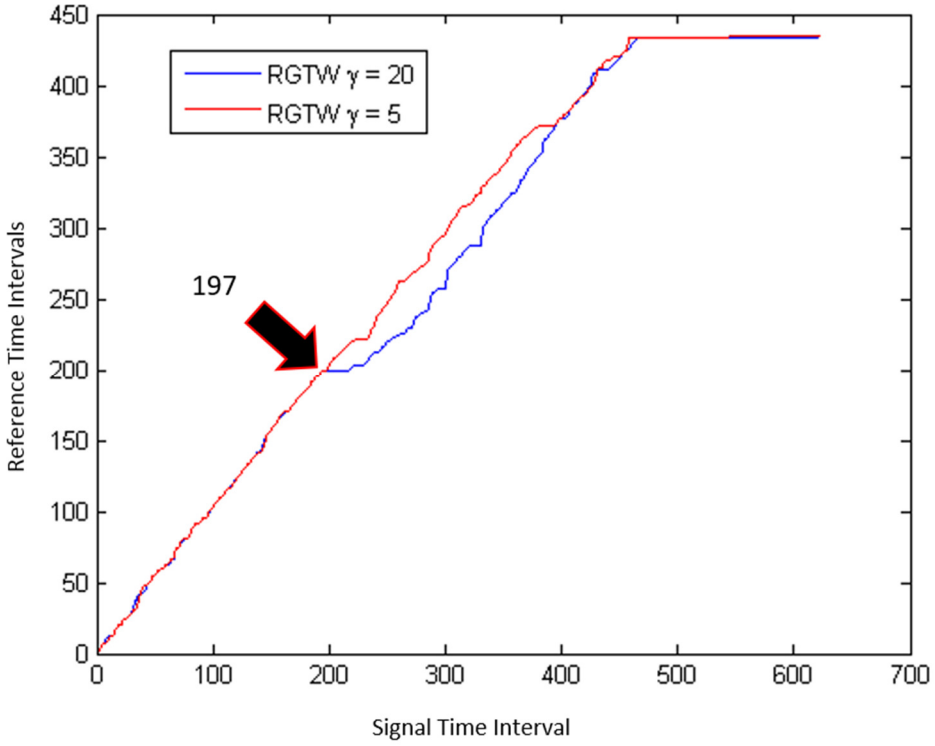


Figure 4.15: Synchronisation Path Deviations for Fault Test Batch

In almost all the alarms detected, only the SPE_x statistic exceeded the control limits imposed. The Hotellings T^2 statistic rarely exceeded its limit. The limit was calculated based on the F-statistic at the confidence level and A and $I - A$ degrees of freedom. This limit is quite high if a smaller number of training batches (I) are used and fewer PCs (A) are retained, which is the case for this simulation. Adjusting these control limits is recommended, as the Hotellings T^2 does increase in some of the cases when the fault occurred, just not enough to exceed the imposed limit (figure 4.16).

Overall, the MPPLS and operational phase models showed the best performance. The false alarm rates were closest to the 5% confidence limits for the NOC batch. However, the detection delay was the longer for these models than the global model, which could be as a result of the confidence limits being high enough to allow the minor number of false alarms. It is inconclusive whether these missing alarms are a result of the nature of the fault itself, although the deviation from the model occurred consistently around the 200^{th} interval for both phase-based models. The number of false alarms recorded in before the fault occurred in the fault batch was much lower for the operational phase model than the MPPLS model, which is due to the lower limits in the second operational phase for the MPPLS model. The limits would probably need to be relaxed slightly in this region. Based on these results, the best monitoring scheme would probably include using the operational phase model with a dynamic window size that changes when tank 2 gets closer to its set point.

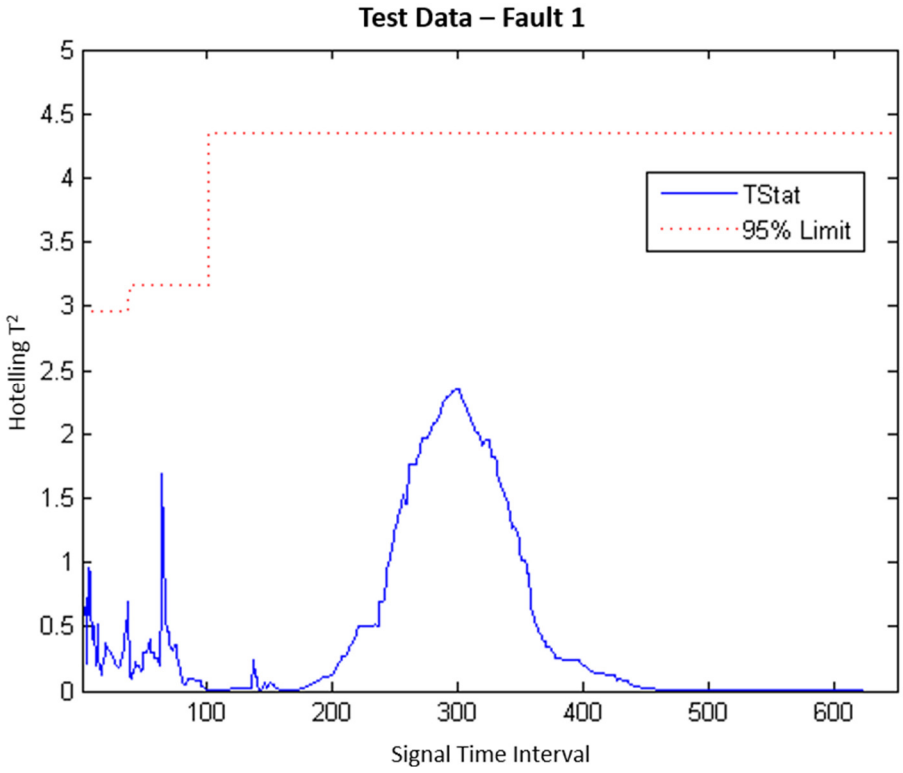


Figure 4.16: T^2 Chart for Fault Test Data (Operational Phase Models, $\gamma = 5$)

4.5.2 On-line End-of-Batch Prediction

The following section presents the results of the ability of the models to predict the end-of-batch quality. Only one variable was required to be predicted, and most of the models trained accounted for a large amount of the variance in the product variable. Figures 4.17 and 4.18 show the deviations of the predictions for the actual quality value at every point along the trajectory. The global model provided the best predictions, although the predictions were reasonably accurate from the 200th interval for all methods. The larger window size offered more accurate predictions, especially in the last part of the batch run. This is due to the suboptimal synchronisation for the smaller window size near the end of the batch.

The MPPLS algorithm also partitioned the data at the beginning of phase 3. However, two PCs were chosen to model the second correlation phase as opposed to the operational phase models, which were modelled with one PC for the third and fourth operational phases. The predictions were slightly better for the smaller window size, but generally under predicted the end-of-batch quality in the latter stages of the process for the larger window size. However, the predictions in the beginning of the process were much more stable than the operational phase models. In general, there is no conclusive evidence that any of the three models outperformed the others in the latter phases of the process. However, the global model was much more stable over the initial phases, this model should probably be used going forward.

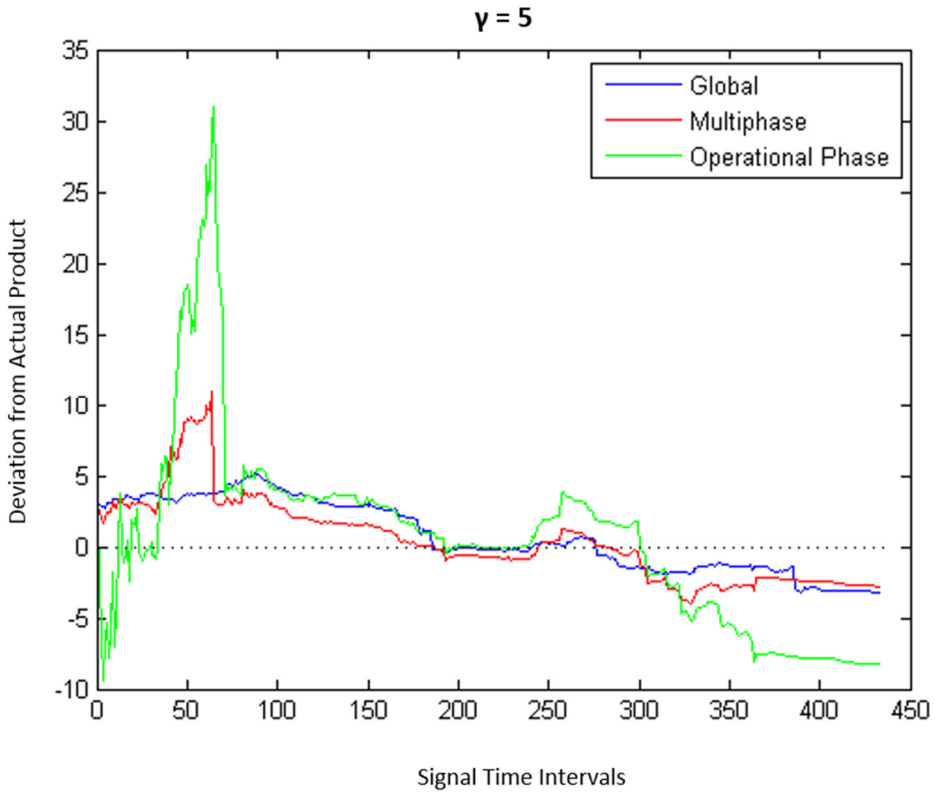


Figure 4.17: Deviation of Predictions from the Actual Quality (γ=5) – Case Study 1

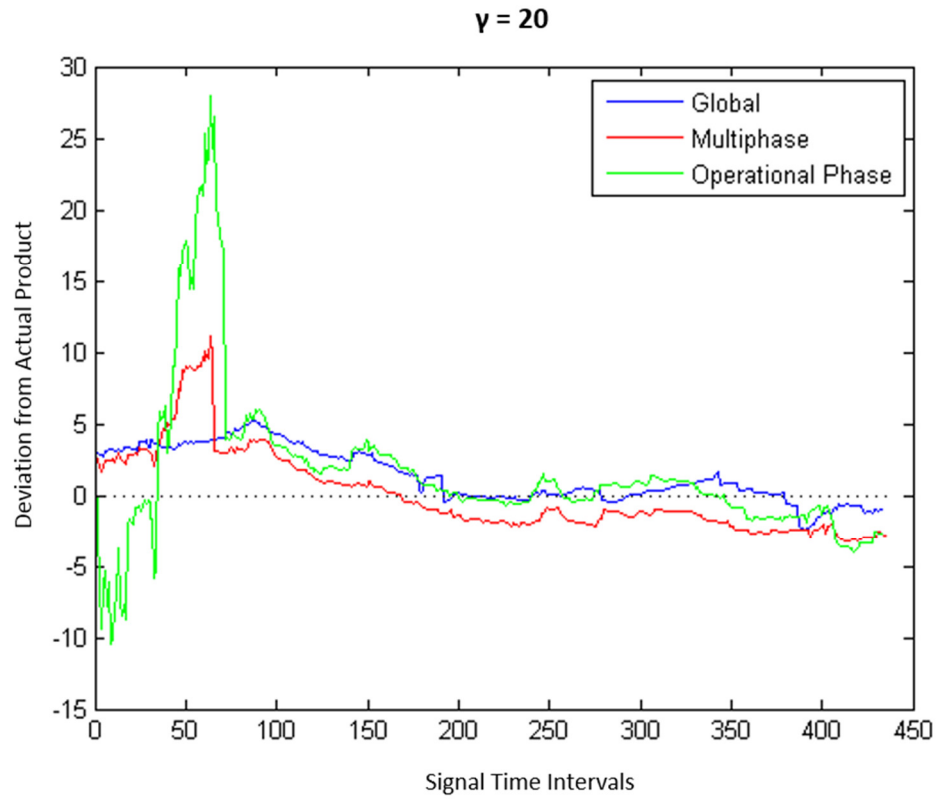


Figure 4.18: Deviation of Predictions from the Actual Quality (γ=20) – Case Study 1

It should also be noted that changing the initial number of PCs parameter (a_{ini}) to 3 PCs in the MPPLS algorithm (section 4.4) returned a global model. This shows that the MPPLS algorithm was able to identify that a global model would make reasonably accurate predictions. No comparison was done between the MSEs obtained via the different initial PCs parameter, but such an analysis could easily be done in practice to assess the differences between the models.

4.6 Summary of Results

4.6.1 Synchronisation

Data were synchronised in an on-line fashion using RGTW with window widths of $\gamma = 3, 5, 10, 20, 50$ and ∞. The PCCs between these approaches and off-line DTW were compared to evaluate the accuracy of warping. Window sizes of $\gamma = 50$ and ∞ showed a statistically significant (p-value > 0.05) improvement in correlation with off-line DTW compared with $\gamma = 3, 5$ and 10

The time taken to synchronise the trajectory at each time interval was also evaluated. RGTW with window sizes of $\gamma = 3, 5, 10$ and 20 all managed to produce results under 1s for every time interval, which was considered sufficient for this process. A compromise was made in accuracy and a window width of $\gamma = 20$ was chosen for on-line monitoring and end-of-batch prediction. RGTW with $\gamma = 5$ was also performed.

4.6.2 Model Training and Phase Division

MPLS model sets were trained for three different approaches. Models were trained globally, per operational phase and per correlation phase using the MPPLS algorithm. All of the models accounted for a high amount of variance in the product quality variable. The MPPLS algorithm returned two correlation phases, split at the 68th interval, which was very close to the operational phase boundary. The first correlation phase was modelled with 1 PC and only accounted for 38% of the total variance in the product quality variable, and the second correlation phase was modelled with 2 PCs and accounted for 99.2 % of the variance in the product quality variable. All three of these models were applied in an on-line fashion for use in monitoring and end-of-batch prediction. Setting the a_{ini} parameter of the MPPLS algorithm returned a global model, which was not used in the analysis.

4.6.3 Model Application for On-line Fault Detection and End-of-Batch Prediction

Models were applied on-line to an NOC batch case and a fault batch for the three models and two window sizes presented above and the false alarms, missing alarms and detection time were recorded and reported on. A large number of false alarms were recorded at the end of the NOC

batch when RGTW was performed with $\gamma = 5$. These false alarms were attributed to improper warping, which is a caveat of using such a small window size. Missing alarms were reported when synchronising the fault batch with $\gamma = 20$, and this was attributed to the fact that time warping over large intervals would distort time information, which was important for the process. Overall, a hybrid approach of applying the window size of $\gamma = 5$ for the first part of the process and $\gamma = 20$ for the second part of the process was recommended to generate accurate results. The operational phase models produced the best performance of the three models.

On-line end-of-batch prediction was performed for all three model sets and the prediction power was discussed. The operational phase models produced accurate predictions from the beginning of the third operational phase. Although the global model performed well in prediction, phase-wise models were recommended because of the superior modelling performance. The MP algorithm produced reasonable results, but was outperformed by the other two models in prediction.

CHAPTER 5 Case Study 2 – Penicillin Cultivation Process

The penicillin cultivation process is a well-studied nonlinear, multistage process. Penicillin-producing cell biomass is first grown in a batch stage, thereafter penicillin is cultivated in a fed-batch stage, feeding glucose into the system at a controlled rate such that penicillin production is favoured over cell growth. A simulation package named PenSim was developed by Birol, Ündey, & Çinar (2002) to model the process and provide a testbed for process monitoring applications, amongst others. The software has been used as a benchmark for testing many batch process monitoring techniques (Doan & Srinivasan, 2008; Ge & Song, 2014; Lee, Yoo, & Lee, 2004; Shen, Ge, & Song, 2015; Ündey, Tatara, & Çinar, 2004; Wan, Marjanovic, & Lennox, 2014).

5.1 Process Description

In the first operational stage, the vessel is charged with filamentous microorganisms and glucose. Biomass growth is favoured in this stage, and when the level of glucose falls below a threshold concentration, the biomass is transferred to the fed-batch stage. In this stage (figure 5.1), glucose is fed as a substrate into the vessel while the pH and temperature are regulated using feedback control. Oxygen is fed into the vessel in the form of air and the vessel is stirred by an agitator.

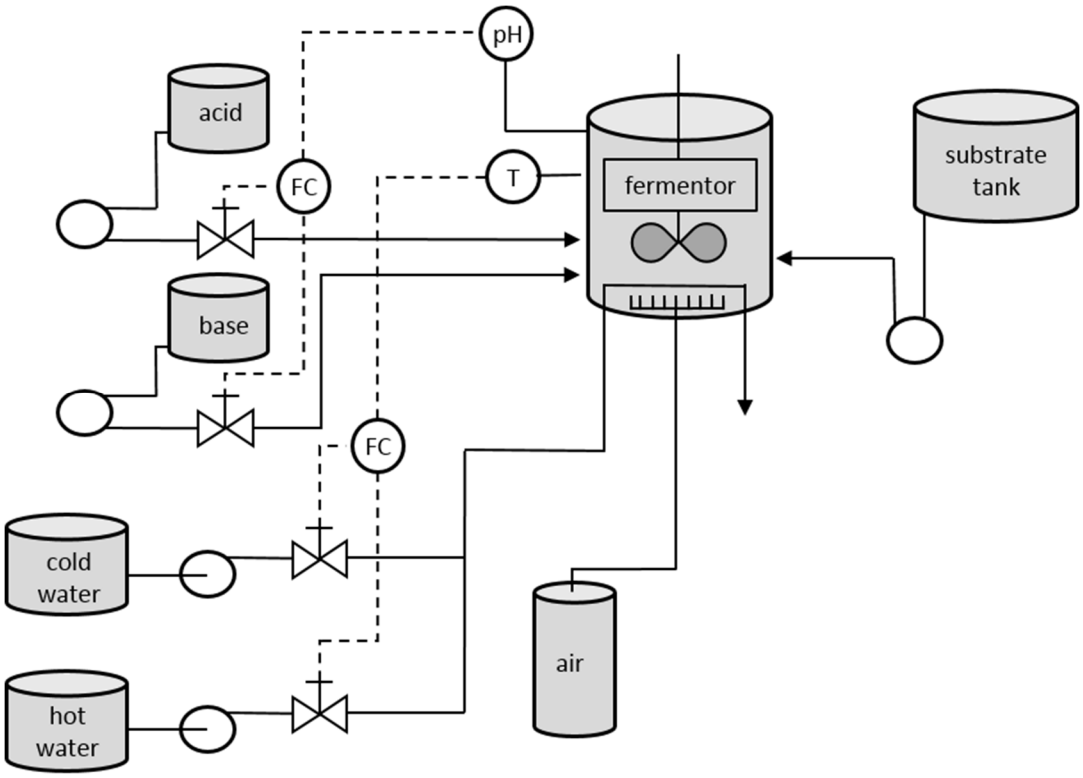


Figure 5.1: Schematic of fed-batch stage of penicillin cultivation simulation (redrawn from Ündey et al. (2004))

PenSim v2.0 (Birol et al., 2002) was used to run the simulations. To introduce variability into the simulation and to mimic the real process environment, Birol et al. (2002) added white noise to the CO_2 and dissolved oxygen profiles to account for sensor sensitivities, as well as a small fluctuation to the glucose feed rate. Small changes in set points and initial conditions were randomly generated within the ranges specified in table 5.1; the same ranges that were used by Lee et al. (2004) to validate their proposed statistical monitoring approach that addressed some drawbacks of conventional MPCA.

The process was said to be complete once 16 L of glucose was fed to the reactor. Simulations were run for 800 intervals, thereafter the data were trimmed to match the criterion for batch completion. Batch lengths varied from 716-800 intervals, with each time interval corresponding to 30 minutes of the penicillin cultivation process.

Table 5.1: Initial Conditions and Set Points used in PenSim

	Variable	Value(s)
Initial Conditions	Glucose Concentration	14-18 g/L
	Dissolved Oxygen Concentration	1-1.2 mmole/L
	Biomass Concentration	0.1 g/L
	Penicillin Concentration	0 g/L
	Culture Volume	100-104 L
	Carbon Dioxide Concentration	0.5-1 mmole/L
	pH	4.5-5.5
	Fermenter Temperature	297-299 K
	Generated Heat	0 kcal
Set Points	Aeration rate	9-10 L/h
	Agitation Power	31-33 W
	Glucose Feed Rate	0.045-0.051 L/h
	Glucose Feed Temperature	296-297 K
	Fermenter Temperature	298-299 K
	pH	5.05-5.15

PenSim recorded the data of seventeen variables over the entire duration of the process. Of these, eleven were used as measured process variables (table 5.2). Time was included as a variable as it was considered important for the fed-batch phase. Synchronisation warps the trajectories and as a result the time information would be lost if it was not included.

Table 5.2: Process Variables used in PenSim

PenSim Variable No.	Process Variables	Units
1	Time	H
2	Aeration rate	L/h
3	Agitation Power	W
4	Glucose Feed Rate	L/h
5	Glucose Feed Temperature	K
7	Dissolved Oxygen Concentration	mmole/L
10	Culture Volume	L
11	Carbon Dioxide Concentration	mmole/L
12	pH	
13	Fermenter Temperature	K
14	Generated Heat	Kcal

Five quality variables were used in accordance with those used by Ündey, Tatara, & Çinar (2004) and are given in table 5.3. The values of the last four quality variables were not directly available from the simulation and were derived from the recorded variables as follows:

$$Overall\ Productivity = \frac{c_{penicillin,final} \times V_{final}}{t_{final} - t_{start}} \quad \textbf{5.1}$$

$$Terminal\ Yield\ of\ Pencillin\ on\ Biomass = \frac{c_{penicillin,final}}{c_{biomass,final}} \quad \textbf{5.2}$$

$$Terminal\ Yield\ of\ Pencillin\ on\ Glucose = \frac{c_{penicillin,final} \times V_{final}}{\sum_{i=1}^{t_{final}} \dot{q}_{glucose,i} \times c_{glucose,i} \times (t_i - t_{i-1})} \quad \textbf{5.3}$$

$$Amount\ of\ Penicillin\ Produced = c_{penicillin,final} \times V_{final} \quad \textbf{5.4}$$

Here, c is the concentration and $\dot{q}$ is the flow rate of a specific component at a specific time index, denoted by the subscripts. The values of time t and culture volume V were also assessed at the relevant time index, denoted by the subscript. Note that the initial glucose present in the biomass growth stage was excluded from equation 5.2.

Table 5.3: Quality Variables used in PenSim

No.	Quality Variables	Units
1	Final Penicillin Concentration	g/L
2	Overall Productivity	g/h
3	Terminal Yield of Penicillin on Biomass	g penicillin/ g biomass
4	Terminal Yield of Penicillin on Glucose	g penicillin/ g glucose
5	Amount of Penicillin Produced	g

Two user-defined phases were chosen as the two operational stages. The phase boundary was identified as the time interval where the glucose feed rate first went above 0.001 g/L. This was indicative of a switch from the batch stage to the fed-batch stage, the threshold limit of 0.001 was used to negate floating point errors in the data from the simulation.

A total of 60 NOC runs were simulated and used as training data, which corresponded to the number used by Ündey et al. (2004). One test batch was generated at NOC, and three batches were generated with the faults listed in table 5.4, as these faults affect the final penicillin concentration (Birol et al., 2002). Similar faults were used by Doan & Srinivasan (2008), Birol et al. (2002) and Lee et al. (2004).

Table 5.4: Faults introduced to PenSim

No.	Fault Type	Time Interval (Sampling Time: 0.5h)
1	15% step increase in substrate feed rate	160-800
2	15% step decrease in substrate feed rate	160-800
3	pH controller failure	0-800

5.2 Batch Trajectory Synchronisation

Due to the variability of batch trajectories during this process, synchronisation of the trajectories was required. Off-line synchronisation could be done in one of two ways: using an indicator variable (IV) or using the off-line DTW algorithm.

5.2.1 Indicator Variable Selection

Undey (2004) used an indicator variable for each operational phase of the process to align the variable trajectories. The variables were derived from the variables measured during the process. For the first operational phase, the percentage volume decrease was used:

$$\%\, Volume\ Decrease = \left(1 - \frac{V}{\left|V_{end,stage\ 1} - V_{start,stage\ 1}\right|}\right) \times 100 \qquad \textbf{5.5}$$

In the second operational phase, Undey (2004) used the percent of glucose fed into the reactor. In this study, the percentage volume decrease was used to align the trajectories of the first phase, but a different indicator variable was used in the second operational phase:

$$\%\, Volume\ Increase = \frac{V}{V_{end,stage\ 2} - V_{start,stage\ 2}} \times 100 \qquad \textbf{5.6}$$

The reason for using a different IV is because the substrate is fed at a constant (albeit noisy) rate throughout the stage. The "percentage volume increase" incorporates both the volume of substrate fed as well as the volume changes due to the process conditions, such as acid/base feed rate and evaporation rate (Birol et al., 2002) and as a result was a more suitable indication of the process.

The indicator variables selected require the knowledge of the final volumes at the end of each phase, therefore can only be used for off-line synchronisation. However, it does provide a reasonable comparison for the DTW and RGTW approaches. In order to provide this comparison, the values of the indicator variables for each batch were mapped to closest matching indicator variable of the reference trajectory selected for DTW. This created a set of path points that could be compared directly to the warping paths (figure 5.2). The IV paths show relatively little compression and expansion of the trajectories, which is understandable considering the linear nature of the volume variable in each phase (figure 5.3).

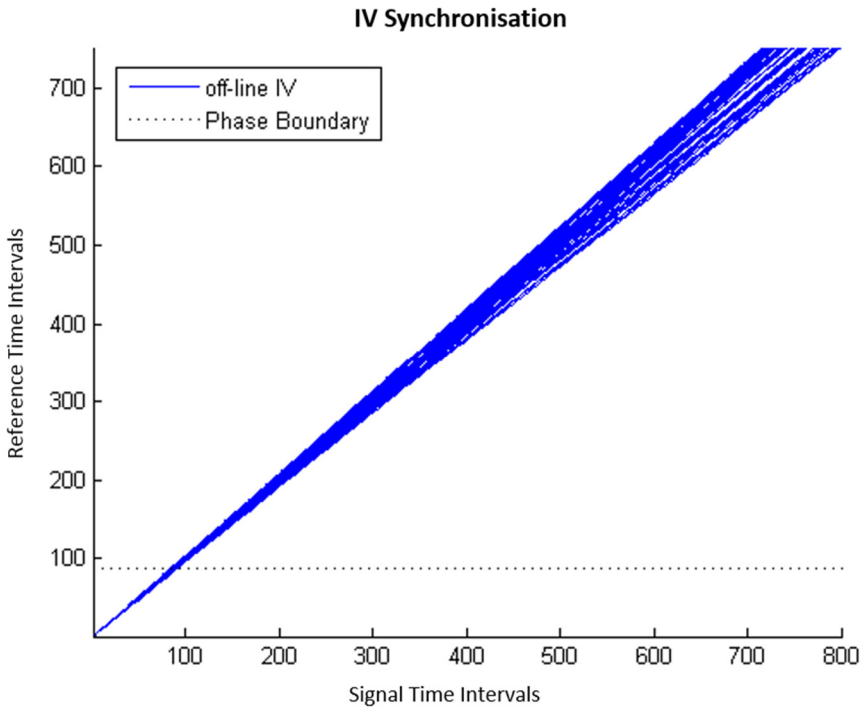


Figure 5.2: Synchronisation Using the Indicator Variable Approach – Case study 2

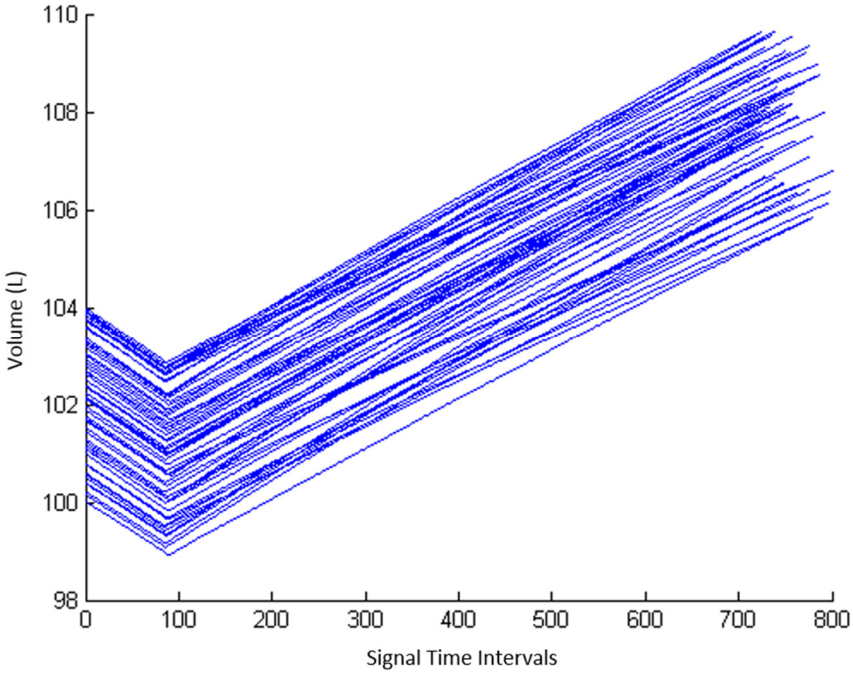


Figure 5.3: Unsynchronised Volume Process Variable – Case Study 2

5.2.2 Off-line Synchronisation

The off-line synchronisation paths of the batch trajectories using DTW are shown in figure 5.4. Off-line DTW showed considerably more expansion and compression of the trajectories than the IV synchronisation, especially in the second operational phase. Most of the variable trajectories are either constant or linear in this phase, so this excessive warping may not be appropriate for this process. Generally, large horizontal or vertical transitions (corresponding with compression and expansion of the signal trajectories, respectively) occur at points along the trajectory. The large vertical transitions at the beginning of the second phase could be the result of a lag phase as the environment changes, but this is unlikely considering that the generated heat variable starts to increase from the beginning of the phase (implying biological activity).

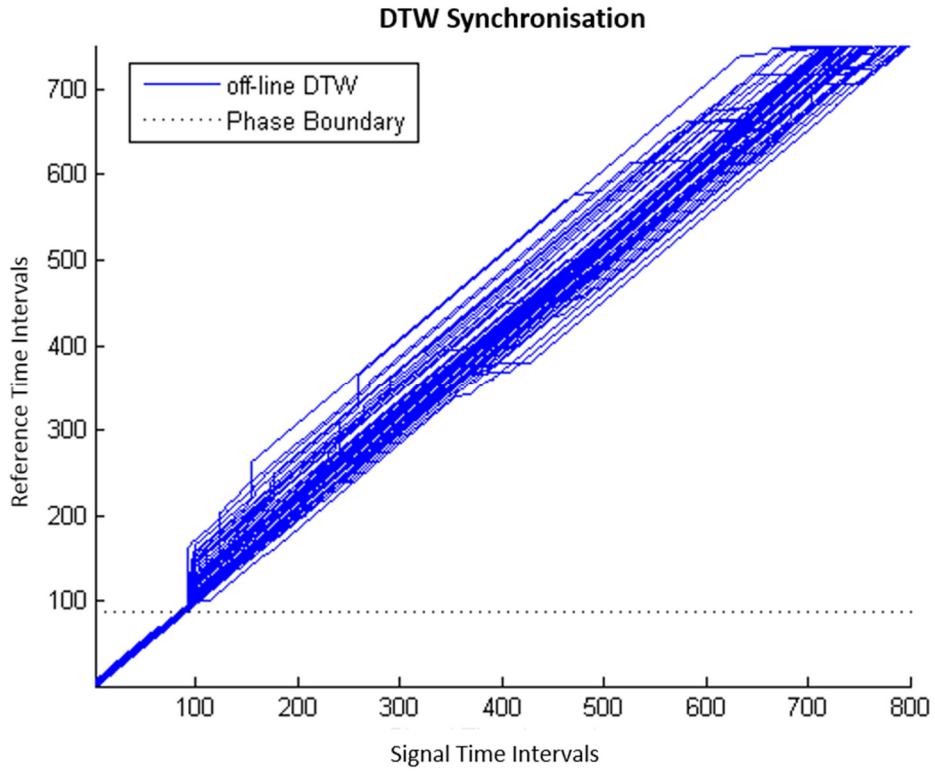


Figure 5.4: Synchronisation Grid Using Off-line DTW – Case study 2

5.2.3 On-line Synchronisation

On-line synchronisation was performed on the NOC data using the RGTW approach with window widths of 3, 5, 10 and 20. The full batch trajectories were also synchronised in an on-line DTW manner by setting the window width greater than the largest time interval in the NOC data (indicated as $\gamma = \infty$). The geometric mean PCCs between the on-line and off-line approaches are shown below in table 5.5. The correlation of the on-line approaches with the IV synchronisation

were higher than with the off-line DTW synchronisation. This is due to the fact that fewer vertical and horizontal transitions are apparent in the approaches with smaller window widths.

Table 5.5: Geometric Mean PCCs for Different Window Widths – Case Study 2

Window Size	Mean PCC (IV, RGTW)	Mean PCC (DTW, RGTW)	No. Samples
3	0.9996	0.9988	60
5	0.9998	0.9988	60
10	0.9997	0.9988	60
20	0.9997	0.9988	60
∞	0.9992	0.9994	60

A plot of the synchronisation paths for $\gamma = 3$ is shown in figure 5.5. Generally, the paths all follow a reasonably linear trajectory, although there is some warping in the first operational phase, as well as a few batches with large horizontal and vertical transitions. Extreme horizontal and vertical transitions are present for one batch in particular in the first phase, which is not accurate.

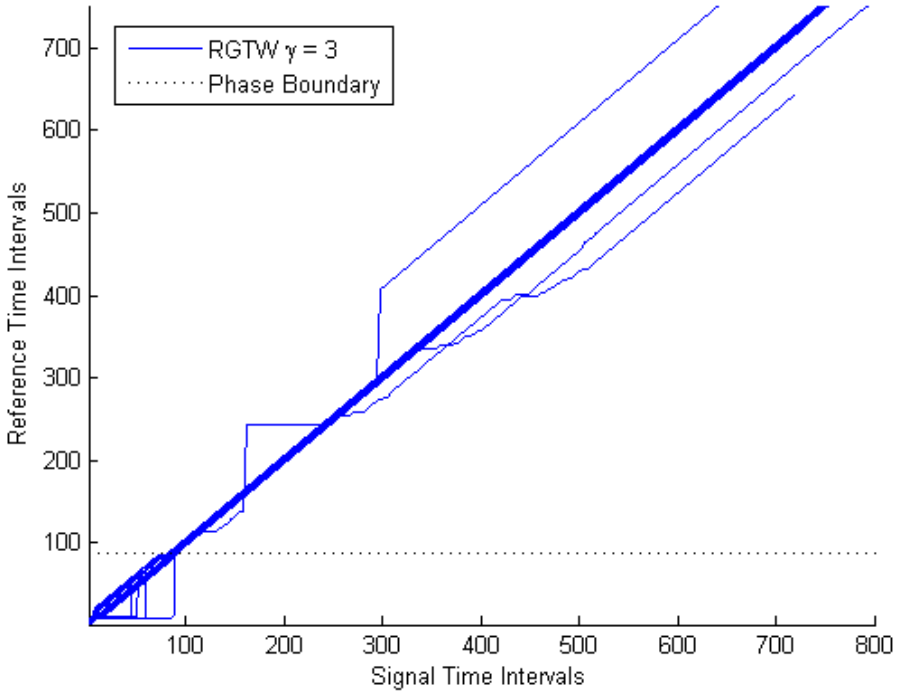


Figure 5.5: RGTW Synchronisation Paths ($\gamma = 3$) – Case Study 2

If the window width is increased to $\gamma = 20$ (figure 5.6), the extreme warping in the first operational phase disappears. Vertical and horizontal transitions are still apparent in some batches, but this is acceptable compared with the DTW synchronisation. However, some expansion and contraction is visible in the first phase, which did not occur in the DTW or IV approaches. The variables were scaled across the entire trajectory in on-line synchronisation as opposed to per phase in off-line warping, so this may have caused the path deviations.

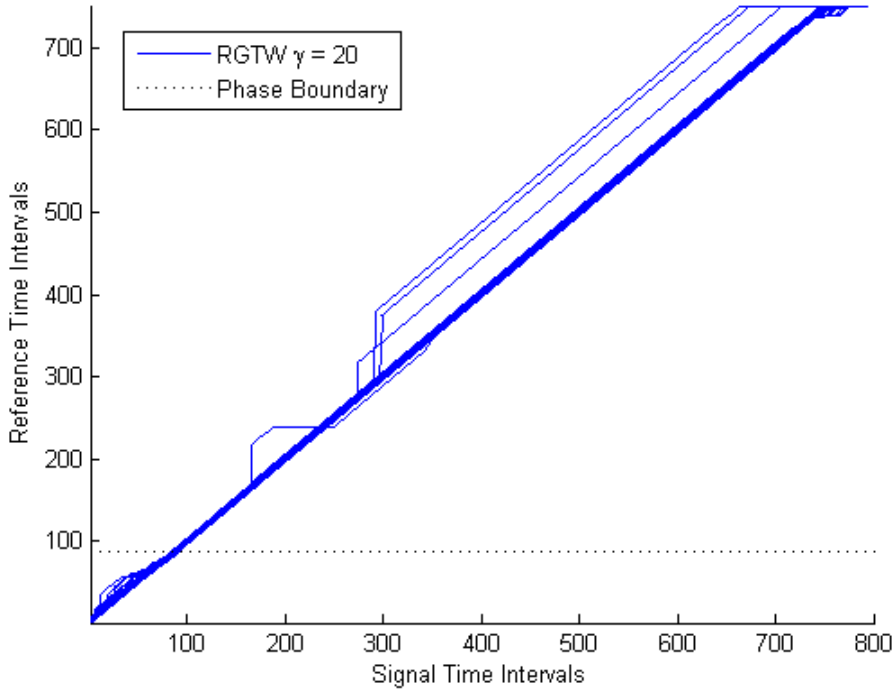


Figure 5.6: RGTW Synchronisation Paths (γ = 20) – Case Study 2

Adjusting the window width to a size greater than the largest batch trajectory synchronises the data in an on-line DTW manner (figure 5.7), which resembles the off-line DTW synchronisation paths more than the other approaches, hence the linear correlation. An ANOVA was performed on the Fisher z-transformed PCCs to test for statistical significance (p-value > 0.05). Figure 5.8 shows that the increased linear correlation of off-line DTW with on-line DTW is statistically significant compared with the other approaches. Conversely, figure 5.9 shows that the decreased correlation of the off-line IV approach with on-line DTW is statistically significant compared with the other approaches.

Off-line DTW was used to synchronise the training data before models could be trained, so it is important to process data in a similar fashion. However, the PCCs between all the approaches is quite high, so most warping approaches would produce reasonable synchronisation over the entire batch duration.

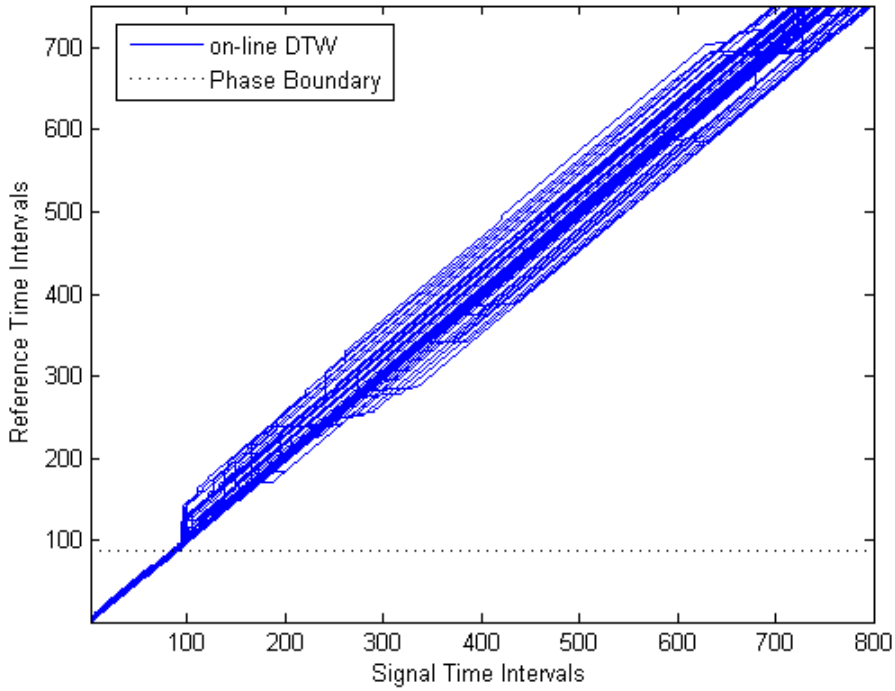


Figure 5.7: On-line DTW Synchronisation Paths ($\gamma = \infty$) – Case Study 2

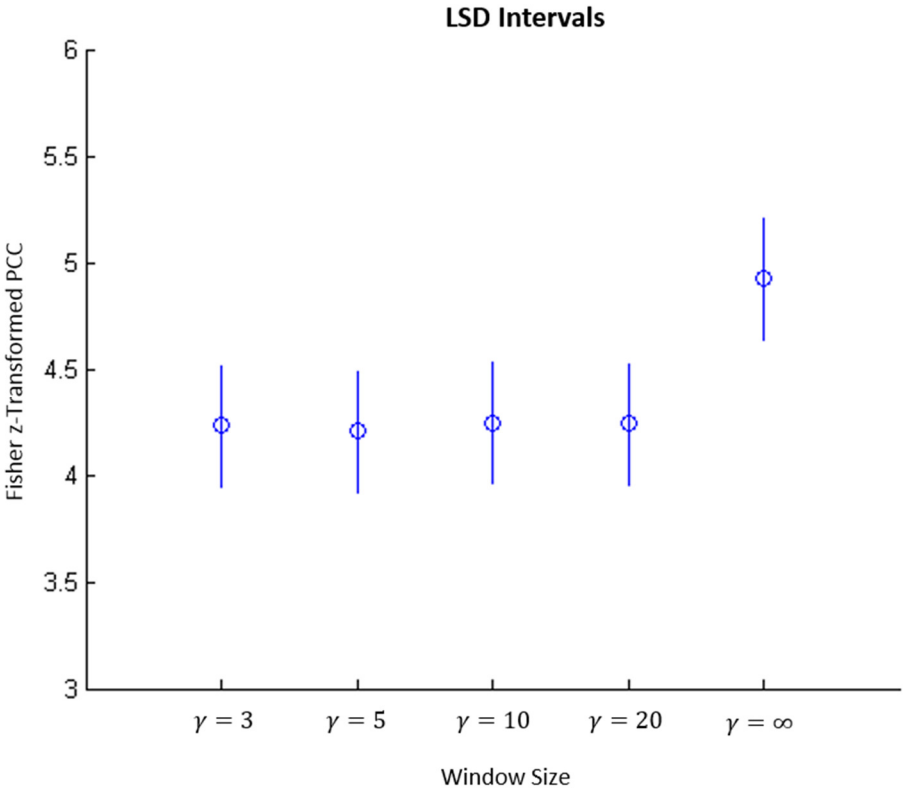


Figure 5.8: Off-line DTW vs On-line Approaches – Case Study 2

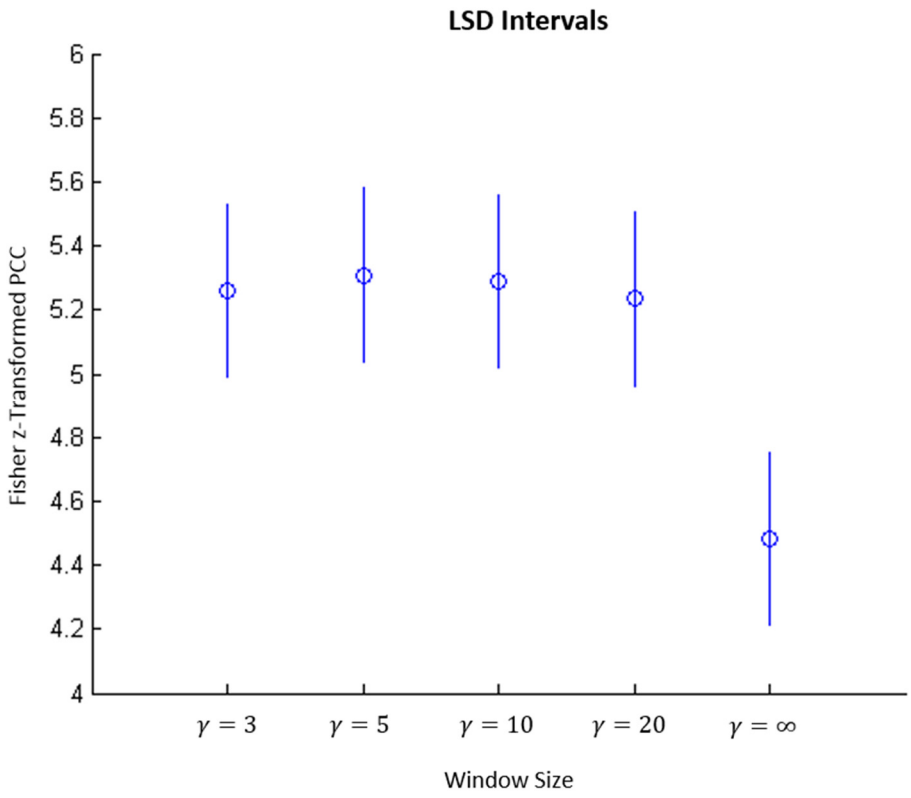


Figure 5.9: Off-line IV vs On-line Approaches – Case Study 2

5.2.4 Computational Expense

The amount of time taken to produce a result for every time step was recorded, and a summary of the information is given here. The on-line DTW approach was only performed on the entire batch trajectory of each batch in the training data. The results show very good computational performance among all the approaches. Measurements are taken at 0.5h intervals, thus all on-line approaches would practically be able to be implemented on-line. However, $\gamma = 20$ was chosen as it offers good correlation with both the off-line DTW and IV approaches.

Table 5.6: Computational Efficiency of On-line Synchronisation – Case Study 2

	$\gamma = 3$	$\gamma = 5$	$\gamma = 10$	$\gamma = 20$	$\gamma = \infty$
Number of samples	45060	45060	45060	45060	60
Average (seconds)	0.0663	0.0727	0.1157	0.2271	103.68
Standard Deviation	0.0319	0.0385	0.0539	0.1002	3.1244
Minimum	0.0024	0.0024	0.0024	0.0025	98.0829
Maximum	0.3611	0.5140	0.9022	0.5731	108.271

5.3 Model Training and Phase Division

Similar to the three-tank simulation case study, three sets of models were trained for application to the data: a global model, operational phase models and a Multiphase (MPPLS) model based on correlation phases. The results of the phase division and modelling are given in table 5.7. The number of principal components (PCs) to retain was done using the cross validation procedure. The global model required only 2 PCs to explain 73.7 % of the variance. 9 PCs were retained for the 1st operational phase, but could only account for 52.0 % of the variance in Y. 5 PCs were retained for the 2nd operational stage, which explained 80.3 % of the variance.

Table 5.7: Data Partitioning and Cumulative Variance Explained for Different Model Sets – Case Study 2

Phase Division	Phase	Reference Interval	No. PCs Retained	Cumulative Variance Explained in Y
Global	1	1 – 751	2	73.7 %
Operational Phase	1	1 – 87	9	52.0 %
	2	88– 751	5	80.3 %
MP algorithm	1	1 – 310	3	71.1 %
	2	311 – 361	3	71.9 %
	3	362 – 411	3	72.4 %
	4	412 – 483	2	72.2 %
	5	484 – 550	3	74.4 %
	6	551 – 751	2	74.1 %

The MPPLS algorithm identified six correlation phases, shown in the flow diagram (figure 5.10). Data were initially split at the 311th interval. At this point in the process, the variable gradients become much flatter, thus the correlation structure may be different. As the process is a biological process, the biological phase could have changed at this point from the growth phase to the stationary phase.

Two more PCs are added to the first phase, while the second phase is further split up (after the addition of a PC). The minimum number of intervals for a feasible split to occur ($minL$) was set to 50 intervals, hence the size of phases 2 and 3. More splits were determined after the 311th interval, which is intriguing because most of the variables tended to flatten out.

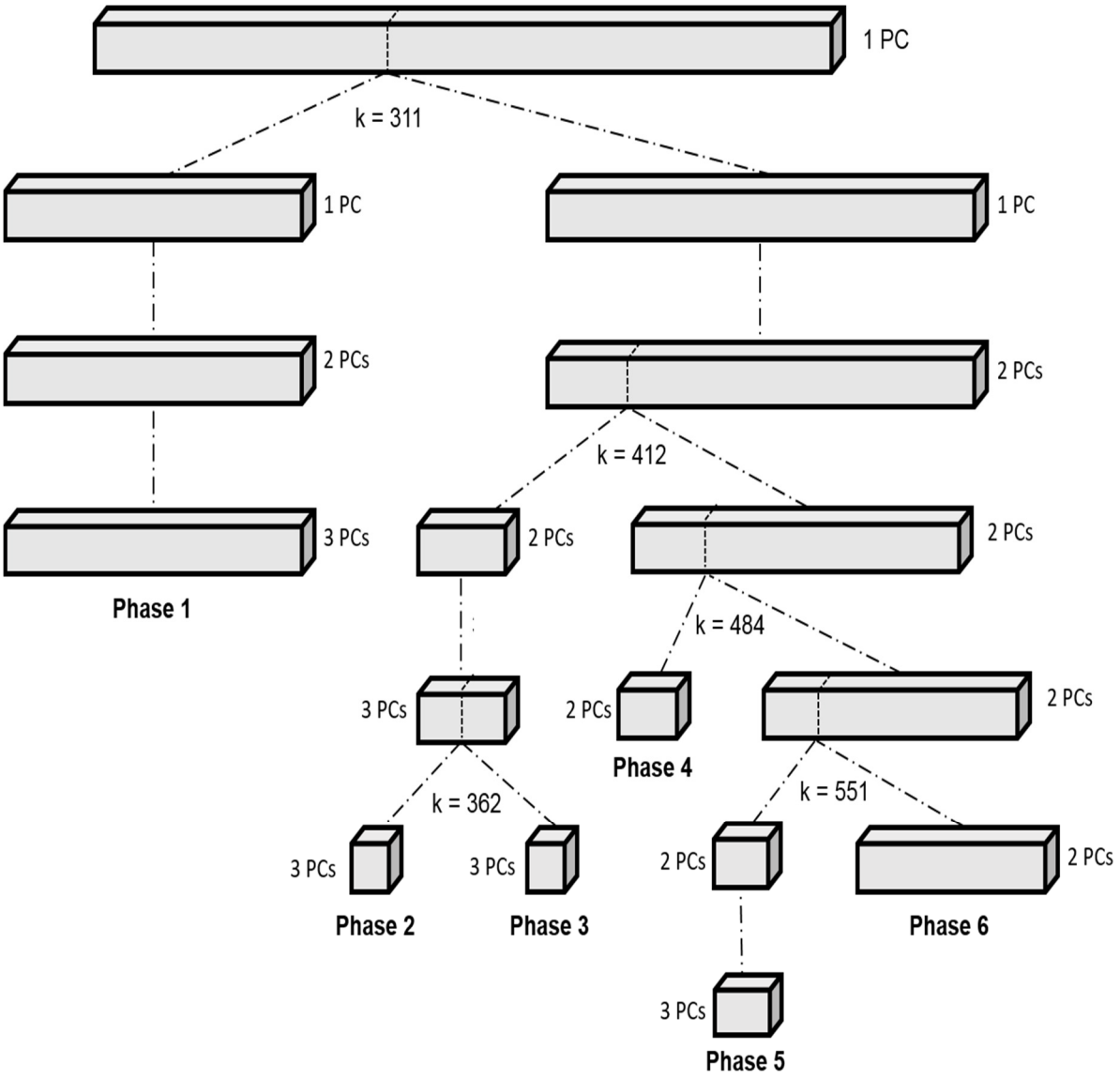


Figure 5.10: Multiphase (MP) Algorithm Data Partitioning – Case Study 2

The MSE for the third correlation phase is shown in figure 5.11. Following the greedy approach, the MPPLS algorithm split the data at the 361st interval after 3 PCs were retained for the 311–483 interval. Consequently, the MSE outcome for the 3rd phase (362–411) was not the lowest. Modelling this phase with 2 PCs would still have been accounted for 71.3 % of the variance and resulted in a lower MSE. This demonstrates how the greedy approach may sometimes produce suboptimal results.

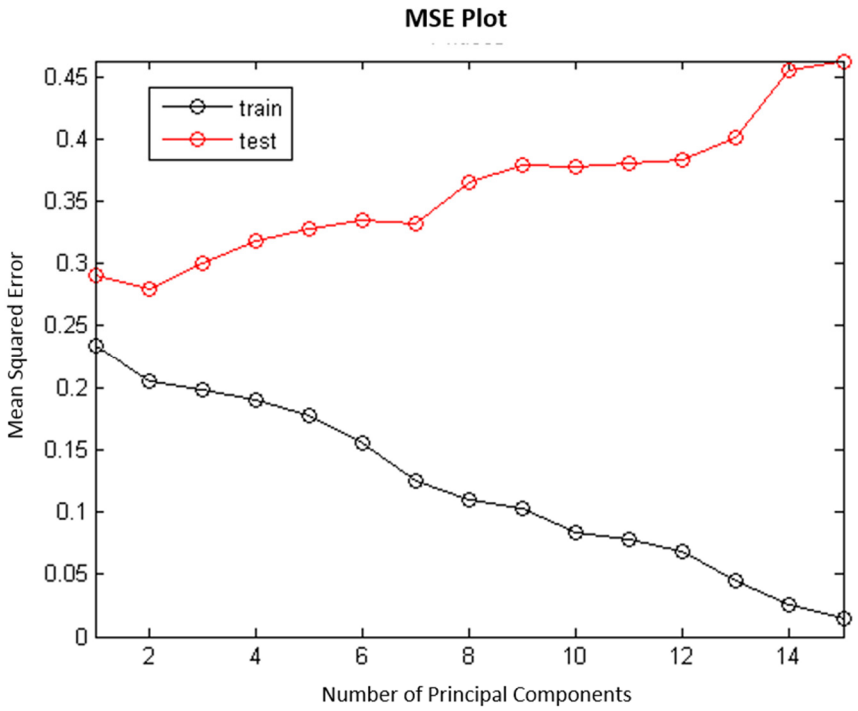


Figure 5.11: MP Algorithm: Third Correlation Phase – Case Study 2

The MSE outcomes of all the models are presented in the figure 5.12. The multiphase model showed a lower MSE in the first correlation phase (1-310). Due to padding the data with zeroes, the MSE of each subsequent phase resulted in spikes.

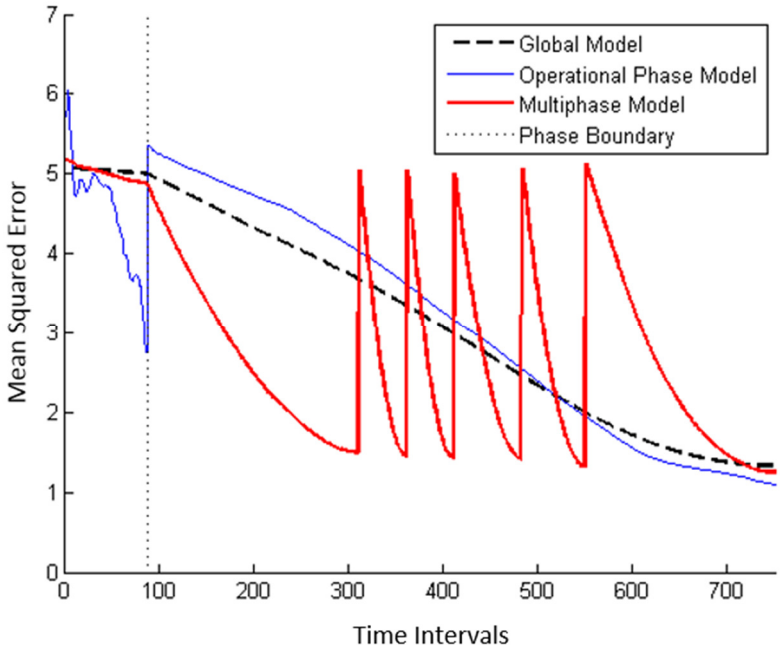


Figure 5.12: MSE Outcomes of Different Models – Case Study 2

The 1st operational phase MSEs exhibit somewhat erratic behaviour over its trajectory. 9 PCs were retained, which only accounted for 52.0% of the variance, so the model may be slightly over specified. An ANOVA test was done (p-value < 0.05) on the MSEs of the three approaches and showed statistically significant reduction of the MSEs for the multiphase model. The MP algorithm uses the MSE as the criterion for partitioning the data, so it is understandable that this would be the case.

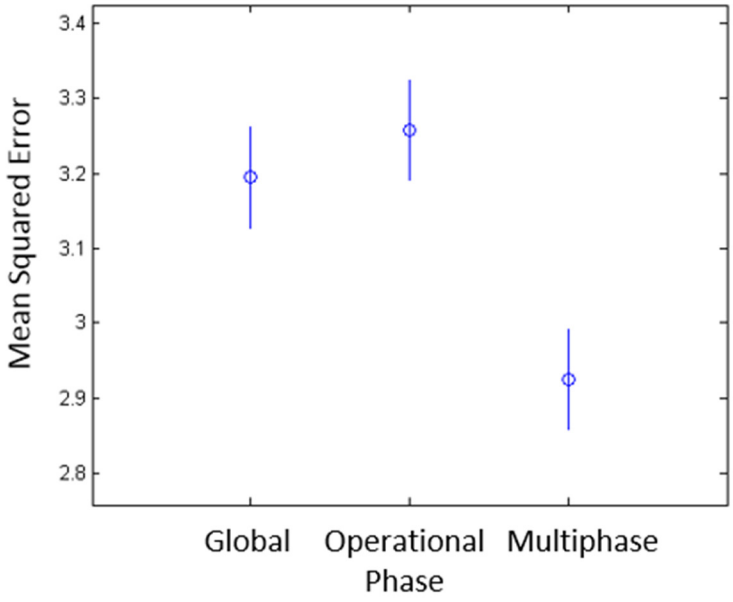


Figure 5.13: Post-hoc ANOVA Test – Case Study 2

5.4 Model Application for On-line Fault Detection and End-of-Batch Prediction

The models trained for the three different phase divisions were applied to the four test cases in an on-line fashion. First, the monitoring results are discussed, then the end-of-batch prediction results are detailed. All monitoring and prediction charts are presented in Appendix C.

5.4.1 On-line Fault Detection

The on-line fault detection results are presented in table 5.8. False alarms were recorded if either the Hotelling's T^2 or SPE_x statistic exceeded the imposed limits for three consecutive intervals. Missing alarms were recorded when neither of the statistics exceeded their imposed limit after a fault had occurred.

Table 5.8: Fault Detection Results – Case Study 1

Test Batch	No. Intervals	Phase Division Method	False Alarms	Missing Alarms	Detection Delay
NOC	789	Global	0	n/a	n/a
		Operational Phases	1	n/a	n/a
		MP Algorithm	2	n/a	n/a
15 % Step Increase in Glucose Feed	668	Global	82	0	3
		Operational Phases	87	4	7
		MPPLS Algorithm	120	0	3
15 % Step Decrease in Glucose Feed	800	Global	25	0	3
		Operational Phases	46	0	3
		MPPLS Algorithm	2	0	3
pH Controller Failure	760	Global	n/a	9	14
		Operational Phases	n/a	7	13
		MPPLS Algorithm	n/a	10	17

The NOC batch run performed very well, with very few recorded false alarms during the batch run. Figure 5.14 shows the SPE_x monitoring chart results for the MPPLS model.

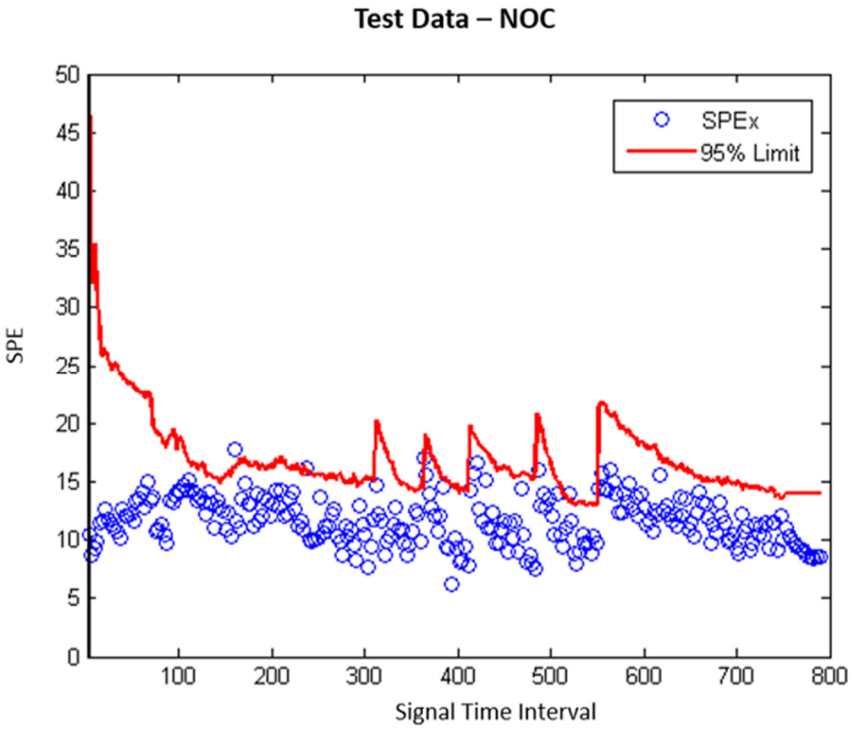


Figure 5.14: NOC Batch SPE Monitoring Chart (Multiphase Model) – Case Study 2

During the first faulty batch run, a significant number of false alarms were recorded. Figures 5.15 and 5.16 show the monitoring charts for the operational phase model. The false alarms occur at the beginning of the batch, during the first operational phase. The synchronisation may have had an influence on the false alarms.

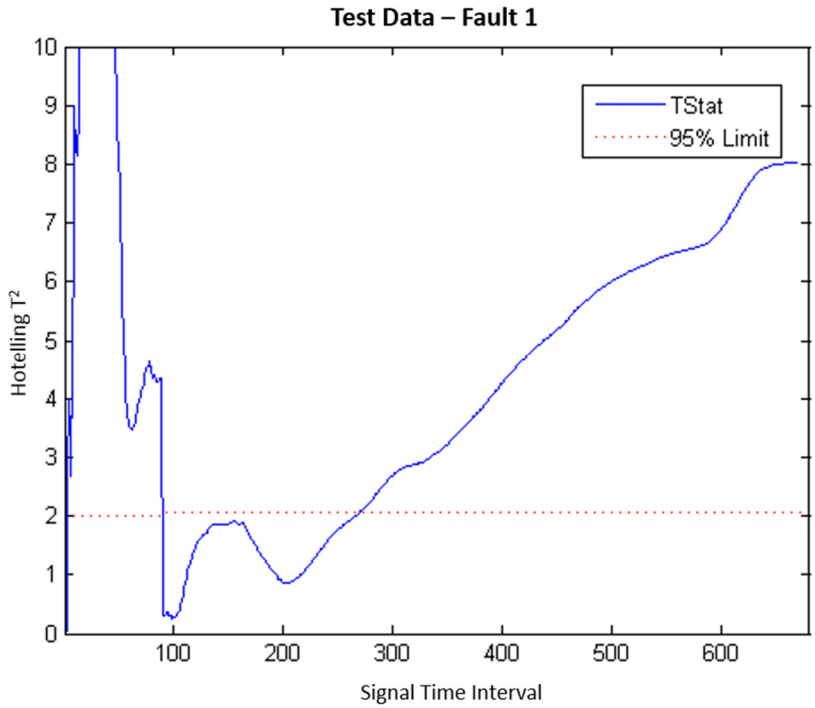


Figure 5.15: Fault 1 Batch Hotelling T^2 Monitoring Chart (Operational Phase Model) – Case Study 2

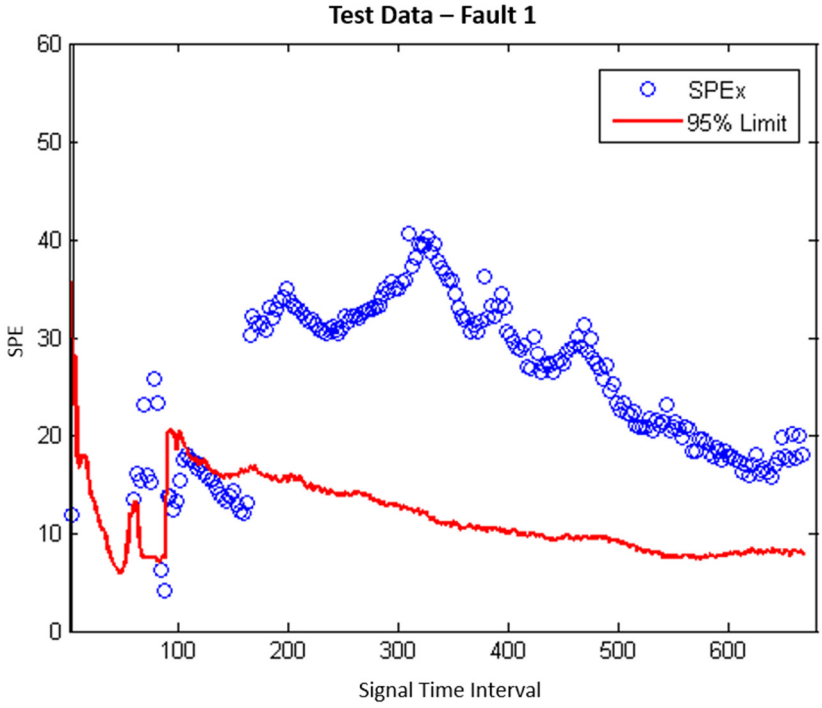


Figure 5.16: Fault 1 Batch Hotelling T^2 Monitoring Chart (Operational Phase Model) – Case Study 2

Figure 5.17 shows the synchronisation paths calculated at every point along the batch trajectory for the first 100 intervals. In the beginning of the batch, the trajectory is synchronised to the incorrect reference intervals. This incorrect mapping would have caused the statistics to exceed their limits and report false alarms.

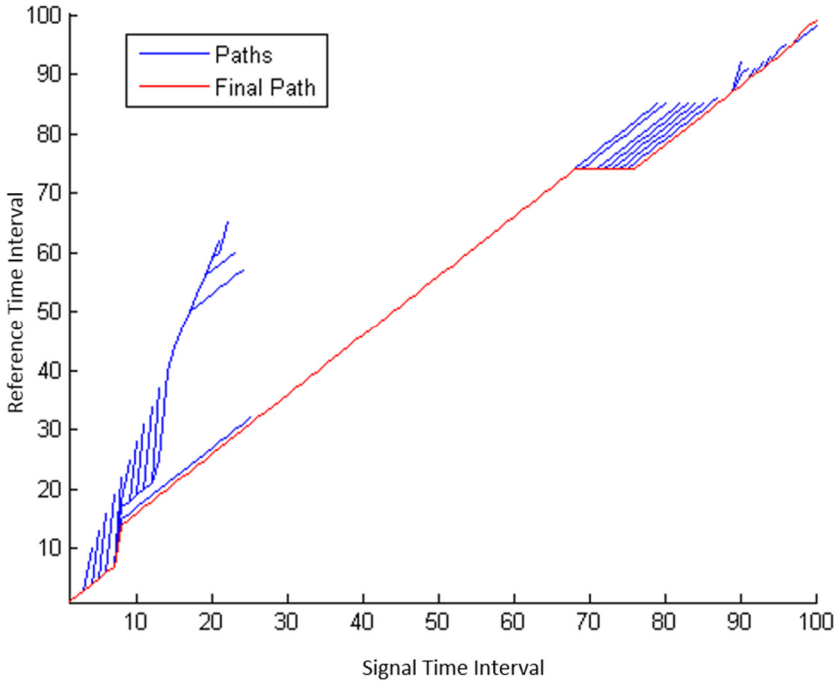


Figure 5.17: Synchronisation Errors During Fault 1 Batch – Case Study 2

The data may not have been synchronised correctly in this phase, as the T_A^2 statistic consistently exceeds the imposed limits for the most of the first phase. The signal interval was expanded at the beginning of the phase and contracted at the end of the phase, which may be inappropriate. These warping inaccuracies were prevalent in the on-line synchronisations as reported in earlier, so this may be the cause of the false alarms. Scaling across the entire trajectory may be an issue that causes such warping problems. No missing alarms were reported for this fault batch, except for the operational phase model. The fault was detected at the 163rd interval, the minimum interval for 3 consecutive alarms to be recorded.

In the second fault case, false alarms occur in the beginning of the batch, as shown in figure 5.17. Additionally, a spike in the statistic is visible at the operational phase boundary. Inaccurate warping here may have caused such false alarms. Extra phase identification may be necessary to alleviate these false alarms. Again, the fault was detected as soon as it occurred, and no missing alarms were recorded in any of the batches.

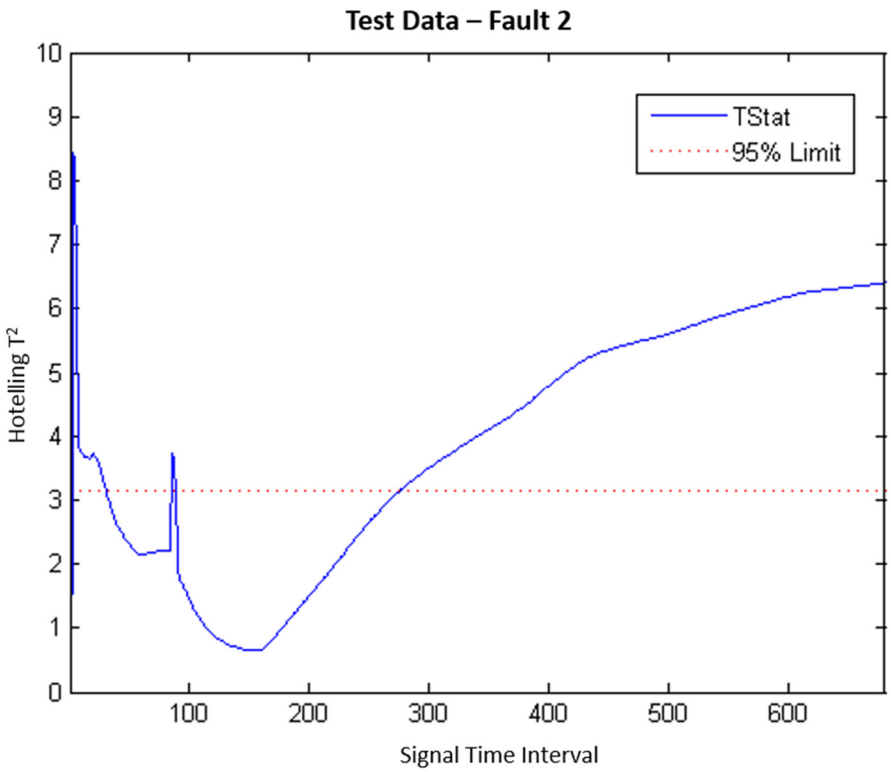


Figure 5.18: Fault 2 Batch Hotelling T^2 Monitoring Chart (Global Model) – Case Study 2

In the last batch run, the pH controller fails from the first time point. The fault is detected after 14-17 intervals. Considering imputing the batch-wise unfolded matrix with zeroes can favour missing alarms in the beginning of the batch, the detection delay is understandable.

Generally, no data partitioning method particularly outperformed the others for fault detection. Synchronisation of the trajectories was responsible for the false alarms in the first phase. Scaling the trajectory in a phase-wise manner for synchronisation may improve the synchronisation accuracy, but this requires some type of proper phase identification. This could be applied to the penicillin cultivation process using the glucose feed rate an indication of the phase transition.

5.4.2 On-line End-of-Batch Prediction

The on-line end-of-batch prediction results are shown for the NOC batch for all of the models. All of the predictions and deviations of the predicted values are given in Appendix C.

Generally, the predictions at the end of each correlation phase were the most accurate for the MPPLS algorithm. The predictions of the global and MPPLS model in the first correlation phase were reasonably similar with regard to Total Amount of Penicillin Produced (figure 5.22), indicating that the first part of the process is important for defining this quality variable. Modelling the operational stages separately did not produce accurate predictions in this region, and it can be argued that the prediction of this quality variable depends both on the first operational stage and

the beginning of the second operational stage. The predictions at the end of this correlation phase are very close to their actual values, and the change of correlation phase provides much more accurate results than the global model, which under predicts the penicillin production from this point onward.

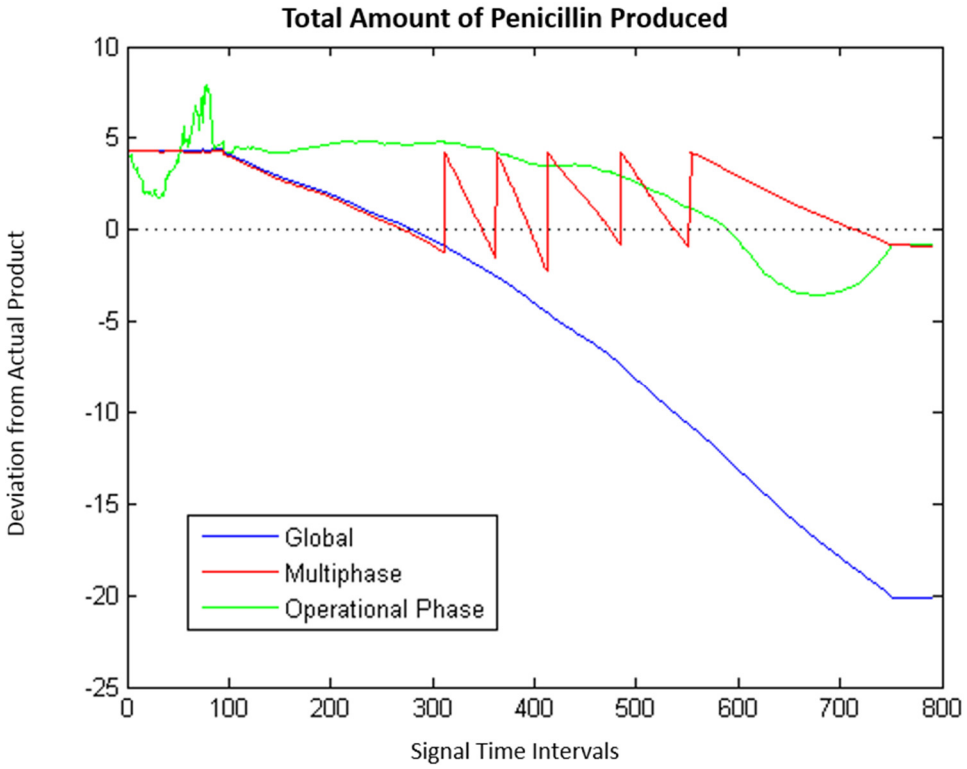


Figure 5.19: Deviations of Total Amount of Penicillin Produced predictions from actual quality – Case Study 2

The predictions of the operational phase model only become accurate very late in the process, and the MPPLS algorithm consistently provides better predictions until the last correlation phase. From this quality variable, it appears that the global model's predictions are influenced by the sharper gradients of the first correlation phase, and cannot provide particularly accurate predictions when the variable gradients flatten and the linear dynamics of the process presumably are much slower. Conversely, the predictions of the second operational phase model account for the slower dynamics of the process to a certain extent, but require a fair amount of the process to have elapsed before the predictions have any value.

In terms of the Terminal Yield of Penicillin on Glucose (figure 5.23), the MPPLS algorithm outperforms both the global and operational phase models in the first correlation phase of the process. The global model generally struggles to make any sort of accurate predictions for this quality variable, and the predictions of the operational phase model from the beginning of the second operational phase to the end of the first correlation phase similarly does not predict anything particularly different from the mean values.

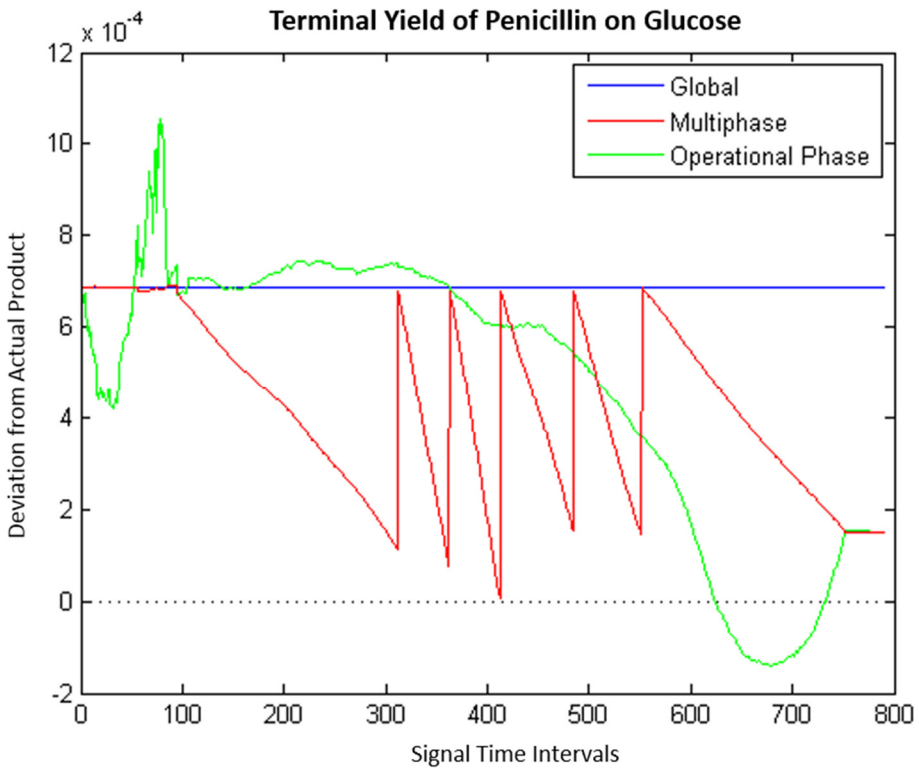


Figure 5.20: Deviations of Terminal Yield of Penicillin on Glucose predictions from actual quality – Case Study 2

5.5 Summary of Results

5.5.1 Synchronisation

Data were synchronised in an on-line fashion using RGTW with window widths of $\gamma = 3, 5, 10, 20$ and ∞. The PCCs were compared to both off-line DTW an indicator variable defined for the process. The on-line DTW approach ($\gamma = \infty$) showed a statistically significant improvement in correlation with the off-line DTW, however it showed the weakest statistically significant correlation with the IV approach.

The time taken to synchronise the trajectory at each time interval was also evaluated. The approaches were all able to produce results in the time required, but a window width of 20 was chosen for application to the models because of its favourable correlation with the IV.

5.5.2 Model Training and Phase Division

MPLS Model sets were trained for three different approaches. The MPPLS algorithm partitioned the data into six correlation phases, and showed statistically significant results in the MSE outcomes when comparing the models.

5.5.3 Model Application for On-line Fault Detection and End-of-Batch Prediction

Models were applied on-line to an NOC batch case and three fault batches for the three models and a window sizes of 20. Almost no false alarms were recorded among any of the batches when the models were applied to the NOC test batch. A high false alarm rate occurred during the beginning of the first faulty batch run, before the fault occurred. These false alarms were attributed to improper warping as the signal points were mapped to many different reference points along each point at the beginning of the batch. Very few missing alarms were recorded for the first and second fault batches, and the faults were detected with almost no detection delay.

On-line end-of-batch prediction was performed for all three model sets and the prediction power was discussed. The MPPLS models outperformed the others, and identified a set of correlation phase that provided superior predictions to the other two methods. Some quality of variables required information from both the first and second operational phase to make accurate predictions, but as the variable trajectories flattened the correlation structure changes, and a new model was needed.

CHAPTER 6 Case Study 3 – Final Concentrate Dissolution Process

Real data from an industrial batch process was obtained to test the ability of the platform to deal with actual data. The data are considered confidential; therefore, all variables were scaled.

6.1 Process Description

In the process, Final Concentrate (FICO) is dissolved using a reagent at high temperature and pressure. Osmium in the FICO is converted to a volatile form (OsO_4). This product is then recovered downstream as potassium osmate (K_2OsO_4) using KOH (Crundwell et al., 2011). Dissolution of FICO takes place through a number of steps. Initially, the required amounts of chemicals to be added are calculated, and reactor is charged. Thereafter, the following steps are executed:

1) **Air displacement** – air in the reactor is displaced with a reagent to prevent a rapid pressure build up during the initial stage of the dissolve
2) **Reactor heating** – the reactor temperature is increased
3) **Dissolve control** – more reagent is fed into the reactor until the desired pressure and temperature is reached. The pressure is increased via addition of the reagent, and the exothermic reaction taking place in the reactor increases the temperature. Once the operating temperature has been reached, cooling water and steam are used to keep the temperature constant. Once the reagent consumption has decreased below a certain level, a sample is taken to analyse the normality and redox potential to confirm the end of the phase. The time taken to analyse this sample is approximately 1.5h.
4) **Cool/depressurise reactor** – The reactor is cooled with air until a certain temperature, and the volatile osmium evacuated until atmospheric pressure is reached.
5) **Osmium sparge** – The temperature is maintained and air is used to sparge the reactor until most of the volatile osmium has been evacuated from the reactor. A sample is taken to determine the osmium concentration remaining in the reactor, once the osmium concentration has fallen below a certain level, the reactor temperature is further dropped.
6) **Boil down** – If the normality of the product is not of sufficient quality, the normality is corrected by reducing the volume and back diluting with water.

This process relies on chemical analysis to determine the quality of the product. If product quality is shown to be inadequate, the batch is reprocessed. Until such time, the reactor cannot be unloaded. It would therefore be advantageous to use the information provided to predict the end-of-batch quality.

6.2 Data Collected during Process

During the process, the following variables were recorded:

1) Reactor Pressure
2) Reactor Temperature
3) Reagent Flowrate
4) Air Flowrate
5) Time

Data were collected from two different reactors and combined for analysis. The batch trajectories of the process variables for one of the batches is shown in figure 6.1. Time was kept as a variable because the synchronisation procedure warps the trajectories, thus this information would be lost if time was not to be included as a variable. Operational phases were chosen as the different steps of the process, indicated by the dashed lines in the figure. The boil down step was excluded from the analysis as it was optional if the normality was insufficient.

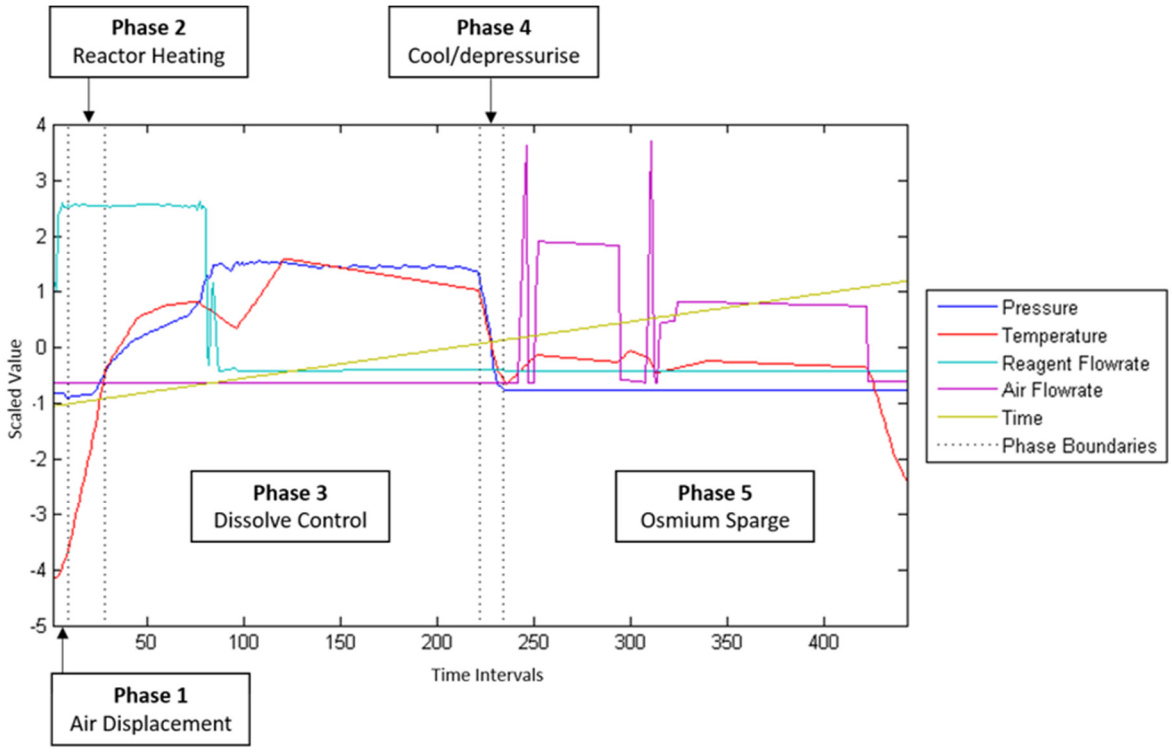


Figure 6.1: Process Variable Trajectories – Case study 3

Three quality variables were identified as important to the process:

1) Final osmium concentration
2) Normality
3) Redox potential

The final osmium concentration in the reactor is a condition for the completion of the osmium sparge and was an intuitive choice as one of the quality variables. Normality and redox potential were used as quality variables to during the process, thus were also included.

NOC data were selected as the batches where none of the steps were repeated. Overall, 45 sets of NOC data were obtained from the two reactors. 39 sets of this data were used for model training, the remaining 6 were left as test batches to apply the models. On analysis, it was found that the quality variables could not be modelled accurately from the available process variables (shown in Appendix D). As a result, this case study could not give meaningful results when used for predictive purposes, including evaluating phase division with the MPPLS algorithm and on-line prediction. However, the process data was still able to provide meaningful analysis for on-line synchronisation.

The batches varied in length from 203 to 975 time intervals, thus significant time warping would be required to synchronise the trajectories to a common pseudo-time axis – much more than in the previous case studies. The two steps most responsible for these time variations are the dissolve control step (phase 3), and the osmium sparging step (phase 5), as the completion of these steps are dependent on the results of specific measurements of certain variables (reagent consumption and osmium concentration, respectively). Figure 6.2 shows the variation between the shortest and longest batch, highlighting the need for synchronisation. Trends can be seen among the batch trajectories of the variables, such as the drop and subsequent rise in temperature in phase 3, and the choice of the reference trajectory used for synchronisation should reflect these trends.

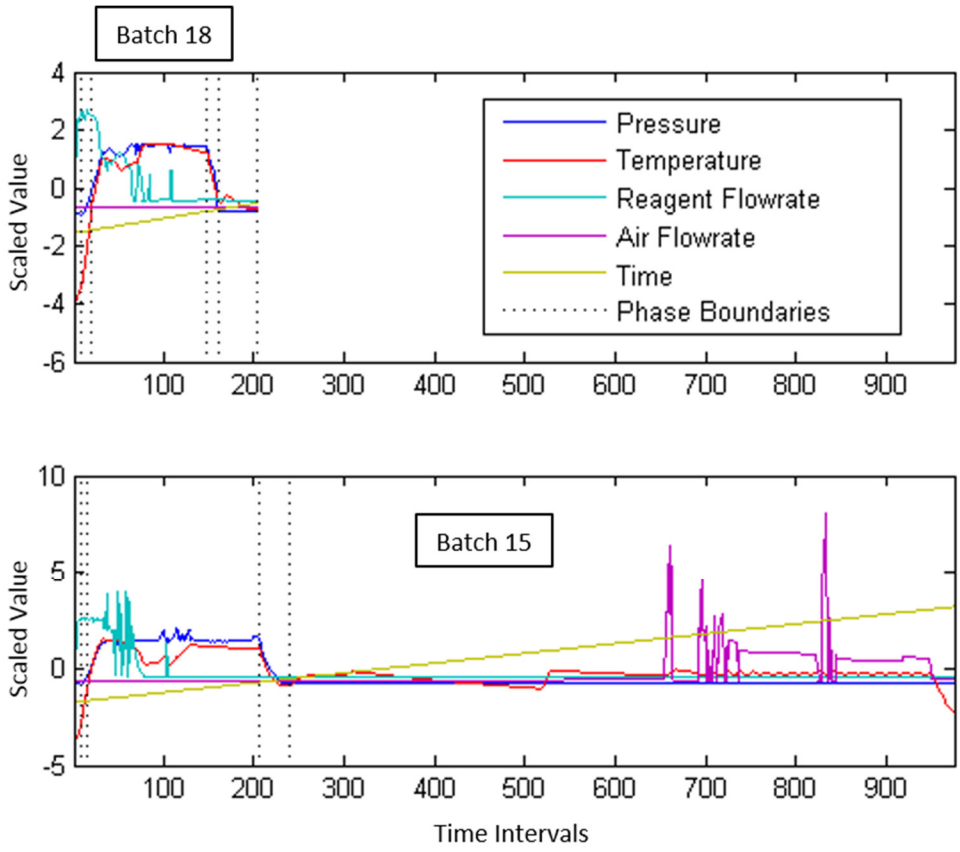


Figure 6.2: Batch Length Variation – Case Study 3

6.3 Reference Trajectory Selection

Data were synchronised using the off-line synchronisation routine, which used Dynamic Time Warping (DTW) to synchronise the trajectories. Reference trajectories were chosen for each phase and the number of time intervals are shown in table 6.1, along with the range of actual time intervals for the NOC data.

Table 6.1: Trajectory Sizes – Case Study 3

Phase	Reference Trajectory Size	Signal Interval Range
1	9	8 – 15
2	12	6 – 24
3	204	129 – 257
4	16	11 – 33
5	279	42 – 736

The reference trajectories are shown in figure 6.3. The first phase is characterised primarily by the increase in reagent flow rate. In the second phase, pressure and temperature are both increasing, and the reagent flow rate is at its maximum. Jumps are also noticeable in the pressure, temperature

and time variables across the phase boundaries, which are due to the concatenation of the reference trajectories across the phases, as each phase was synchronised separately. In the third phase, the trajectories exhibited the most features in the first half of the phase and are relatively constant after the 110th interval. In the fifth phase, the air flow rate variable exhibits step increases in the 264th and 351st intervals. The actual trajectories of most of the batches for this variable are not nearly as constant, as the air flow rate is characterised by spikes rather than a smooth constant trajectory. The reference trajectory was chosen based on the maximum correlation with the trajectories of the NOC data from which it was trained, batch 39 exhibited this constant performance. It is therefore understandable that this batch would show the maximum correlation with the rest of the batches in this phase.

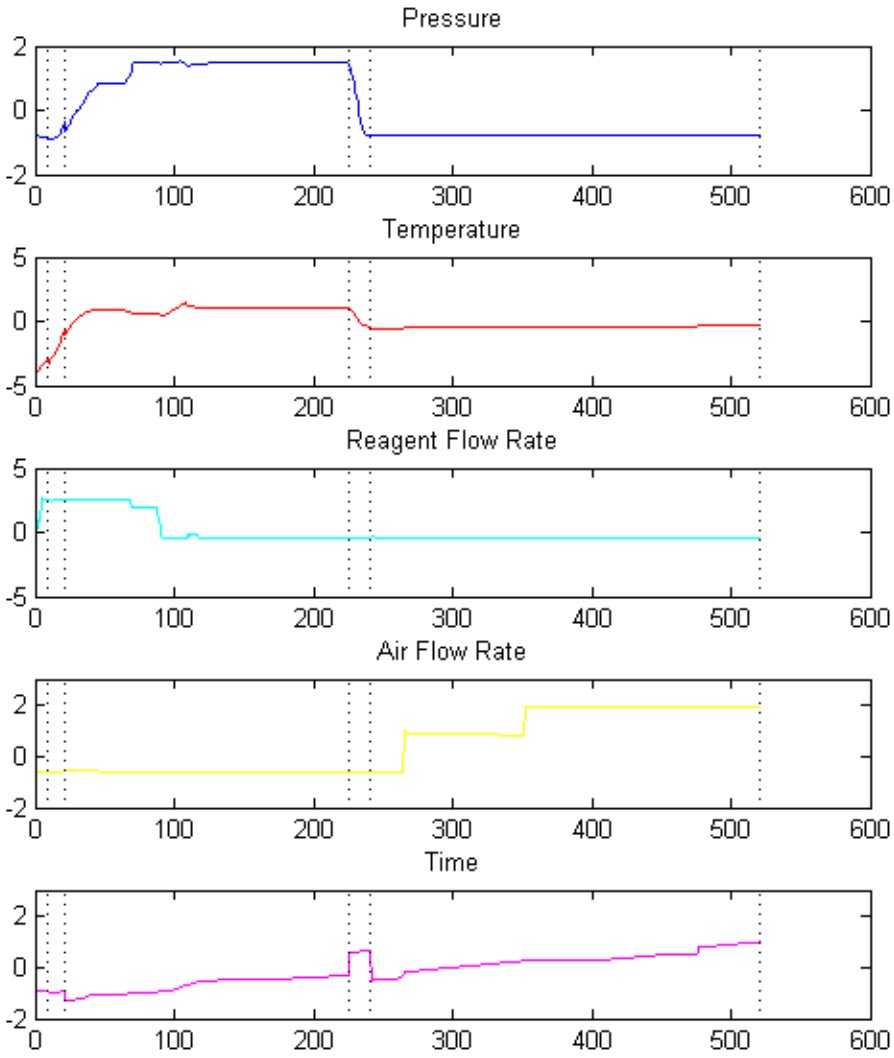


Figure 6.3: Reference Variable Trajectories– Case Study 3

6.4 Off-line Synchronisation

The results of the off-line synchronisation are shown in figure 6.4, along with the reference trajectory phase boundaries. As expected, the most significant warping happened in phases 3 and 5. In phase 3, there were significant horizontal transitions between the 100 and 110th reference interval, indicating a compression of the signal trajectories. Inspection of the warped signal trajectories for pressure, temperature and air flow rate (figure 6.6) indicate that most of the features of the variable trajectories occur in the beginning of this phase, this is due to the pressure and temperature being raised to their operating points. If any of the batches took longer to reach their operating points than the reference batch, these trajectories would have to be compressed in this region. The attempt to warp the feature-rich regions of the trajectory results in a subsequent expansion during part of the phase with relatively constant trajectories. Potential incorrect mapping of the trajectory may be a culprit, as discussed by Gins et al. (2012) when proposing their Hybrid Derivative Time Warping technique.

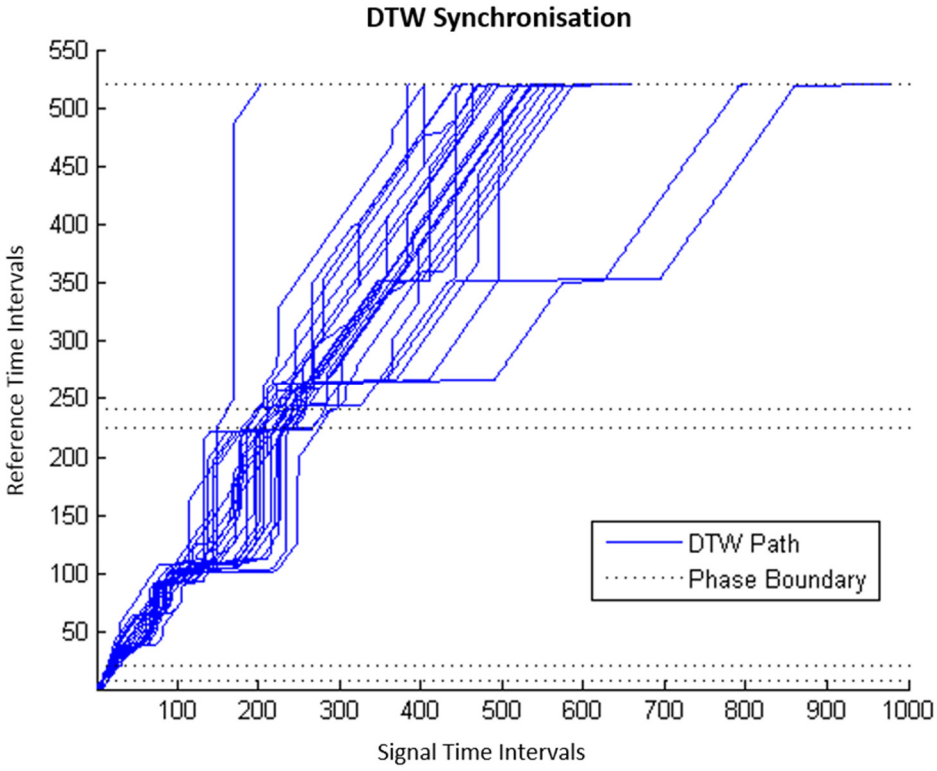


Figure 6.4: Off-line Synchronisation – Case Study 3

In phase 5, the compression of the longest trajectories coincides with the step increases in air flow rate of the reference trajectory. This variable has a significant impact on the warping of the trajectories, and the lack of air flow in the shortest batch (batch 18) results in a large expansion of

the trajectory from the beginning of the phase. This appears to be accurate as the temperature decrease in the osmium sparge step occurs as the batch has been completed (indicated by the decrease in temperature at the end of the trajectories in figures 6.5 and 6.6), and the synchronisation aligns this part of the phase.

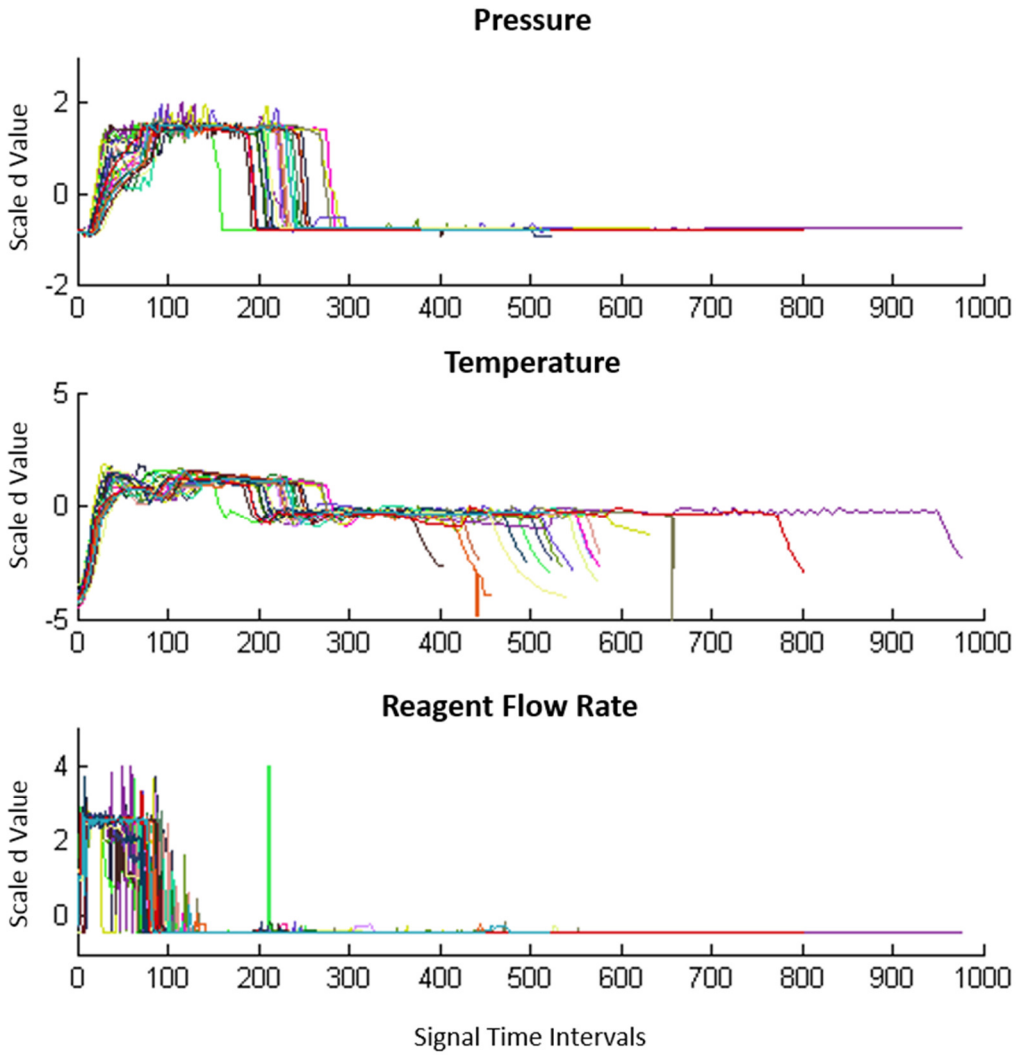


Figure 6.5: Unsynchronised Trajectories of Variables 1-3 – Case Study 3

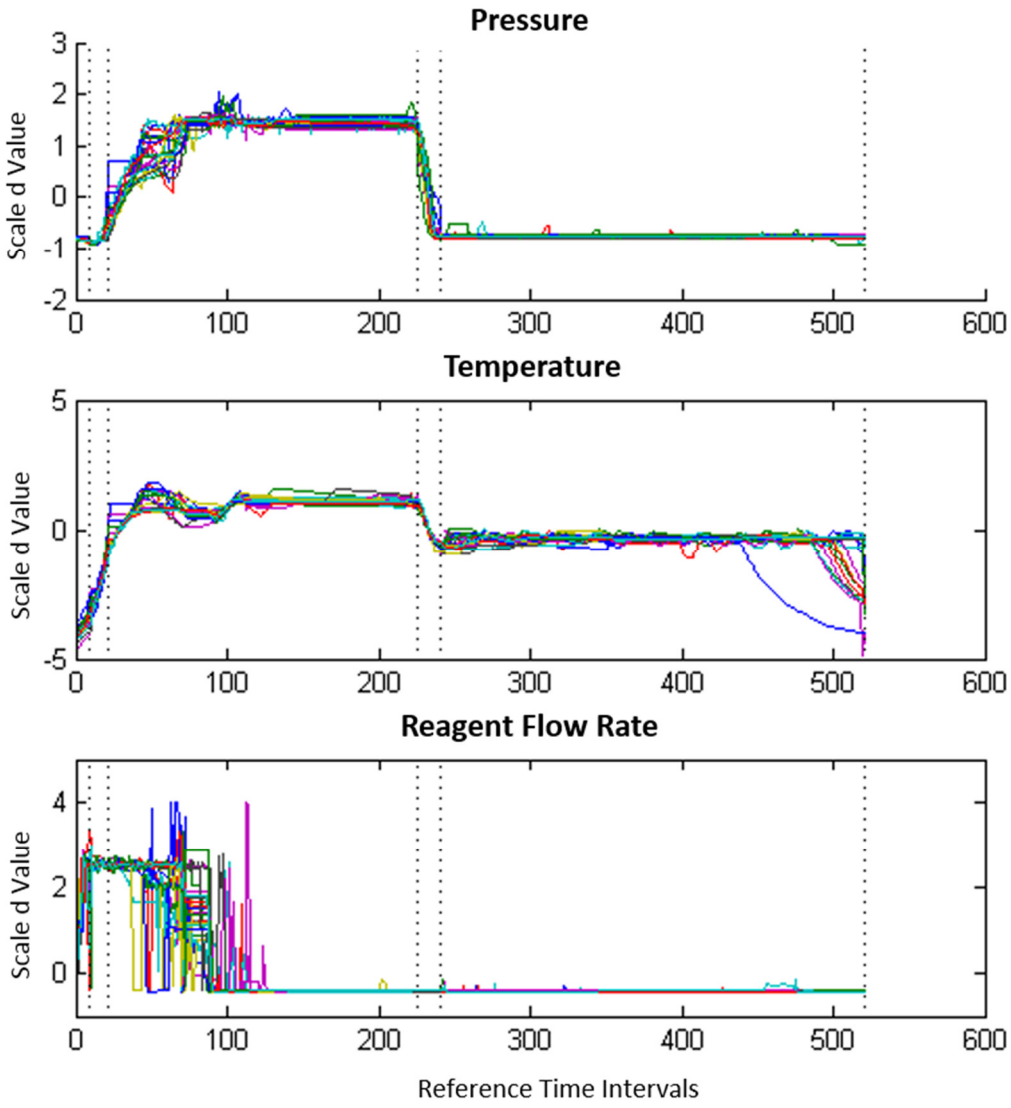


Figure 6.6: Synchronised Trajectories of Variables 1-3 – Case Study 3

6.5 On-line Synchronisation

The off-line synchronisation was performed in a phase-wise manner based on the known phase boundaries, which improves the accuracy of the synchronisation and decreases the overall time taken to process the results. However, no global synchronisation was performed off-line, as increasing the search space for the optimal paths would be computationally expensive. Generally, on-line synchronisation was performed globally as the phase boundaries are usually unknown prior to pre-processing the data on-line. However, in this process, different stages were determined based on operating procedure and as a result the data could be synchronised in a phase-wise manner using expert rules to identify the different phases. Data were synchronised on-line first in a global manner, then in a phase-wise manner. The overall accuracy and computational speed of the two approaches are presented below. The accuracy of the synchronisation was assessed using the Pearson's Correlation Coefficient (PCC), which measured the linear correlation between the synchronised path obtained using the on-line synchronisation methods and between the synchronised path obtained using the off-line synchronisation methods

6.5.1 Global Synchronisation

The results of the global on-line synchronisation using the Relaxed-Greedy Time Warping (RGTW) approach for a window of size $\gamma = 5$ is shown in figure 6.7. General trends apparent in off-line synchronisation are visible here, such as the major compressions and expansions in phase 3. In phase 5, large horizontal transitions occurred near the 352nd reference time interval, which corresponded to the second step increase of the air flow rate variable of the reference batch. This was sometimes followed by large vertical transitions occurring which prematurely expand the signal trajectories at certain time points to the end of the reference batch, as opposed to the gradual transitions exhibited in off-line synchronisation. This most likely is a result of the erratic nature of the reagent flow rate variable (figure 6.8) in most of the batches compared with the flow rate of the reference batch. The algorithm tries to warp an erratic flow rate to a constant flow rate. If the variable fluctuates closer to the second step increase at time at the 352nd reference time interval, the algorithm may incorrectly warp to this point. If the variable stays closer to this value across the entire window size, the warping outside of the window would be permanent and a large transition would occur. This is an understandable caveat of using such a small window size as the algorithm cannot correct the path outside of the window. Conversely, the RGTW algorithm did not warp the endpoint to the reference batch endpoint in 13 (33%) of the batches synchronised. Generally, these batches were compressed significantly at the 352nd reference interval. Instead of expanding the trajectory significantly, diagonal path transitions were preferred.

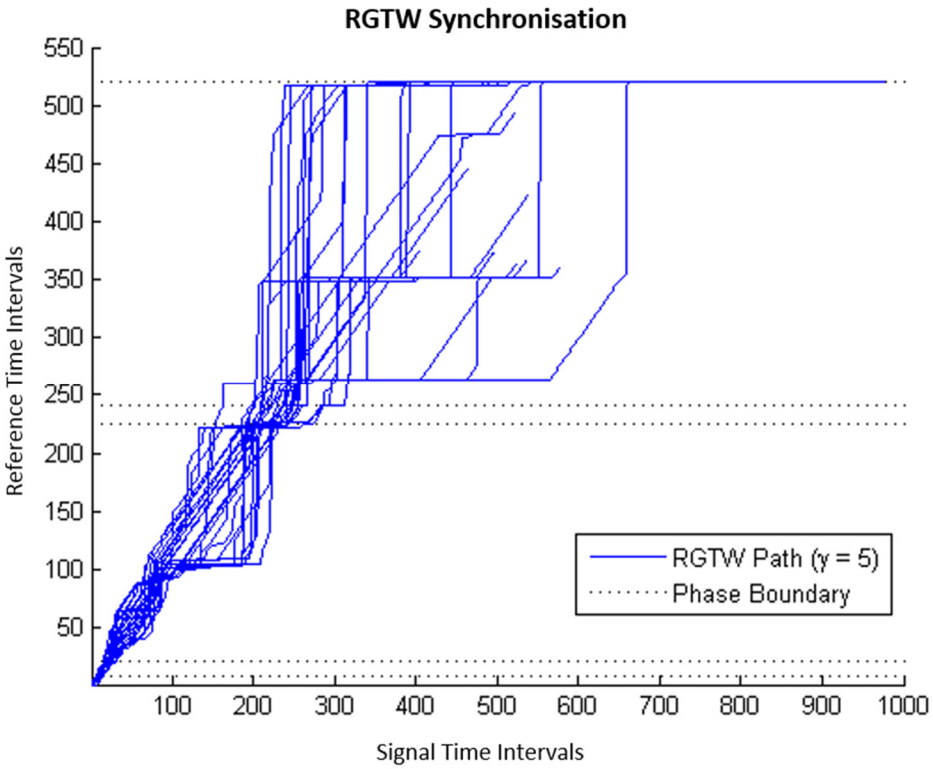


Figure 6.7: On-line RGTW (γ = 5) Synchronisation – Case Study 3

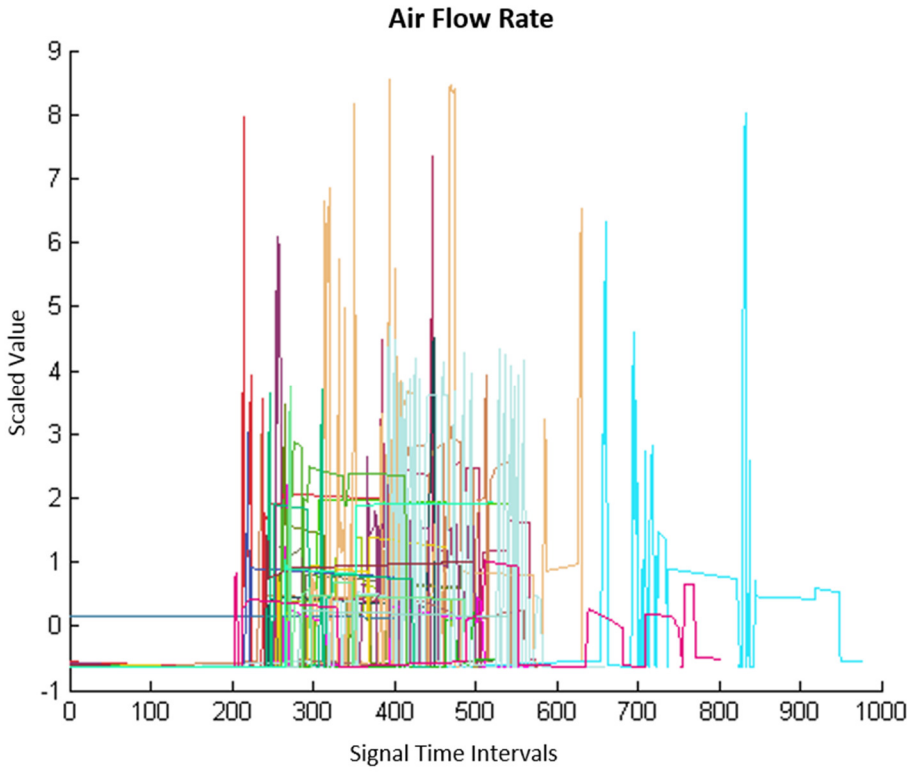


Figure 6.8: Erratic Air Flow Rate – Case Study 3

Global synchronisation was run with $\gamma = 10$ (figure 6.9), and similar results were obtained. The endpoint constraint of the off-line DTW algorithm restricts the path to finish at the final reference interval, and the paths are warped appropriately. Large vertical or horizontal transitions can warp the trajectory within the confines of the overall progress of the batch. In the on-line case, however, the final endpoint is not known. Instead, a series of potential endpoints are chosen and the endpoint corresponding to the path with the least cumulated distance is assumed to be the optimal path.

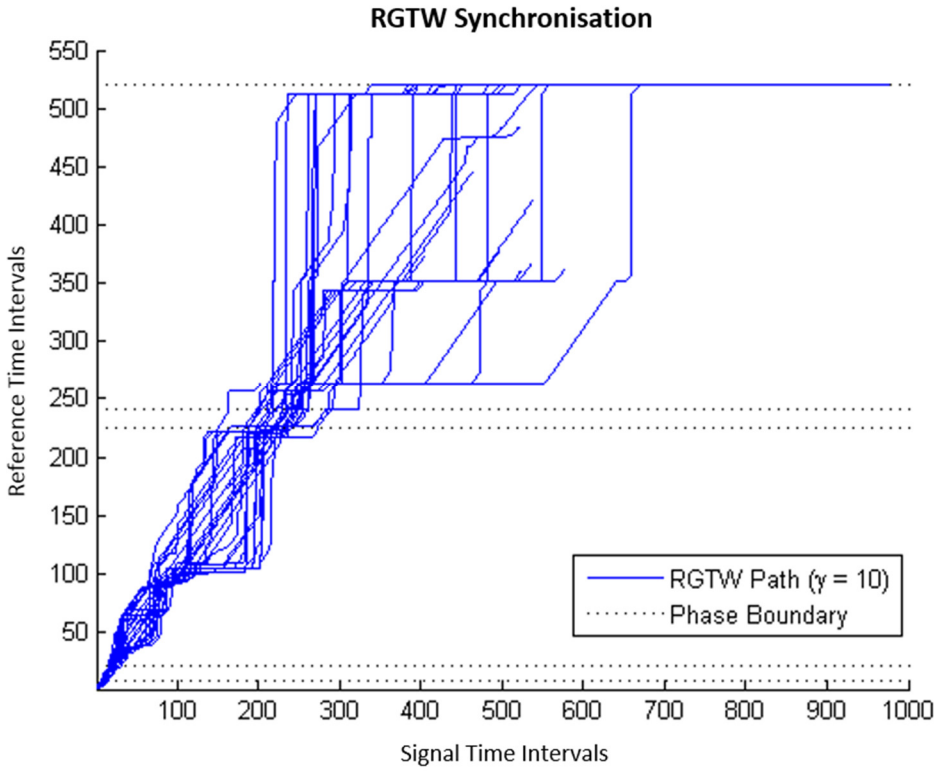


Figure 6.9: On-line RGTW (γ = 10) Synchronisation – Case Study 3

In order to evaluate the differences between off-line and on-line synchronisation, the window size was set to a value larger than the largest signal interval ($\gamma = \infty$). This procedure operates the same way as the Kassidas et al. (1998) approach for on-line DTW. The potential endpoints of the paths were set by the limits during the training phase, these included the actual endpoints of each batch and the results are shown in figure 6.10. The warping generally performed better in the first four phases, but actually warped fewer batches (41%) to their final endpoint than the RGTW algorithm (67%). In the case of the shortest batch (which was warped appropriately off-line), the on-line synchronisation algorithm restricted the final point close to the beginning of the 5th phase of the reference trajectory. Without the endpoint constraint, the on-line synchronisation algorithm

did not recognise the completion of this batch. The completion of the batch occurred with significantly less sparging than the previous batches. The algorithm expected an increase in air flow, thus synchronised the batch to a point before the increase in air flow rate of the reference batch.

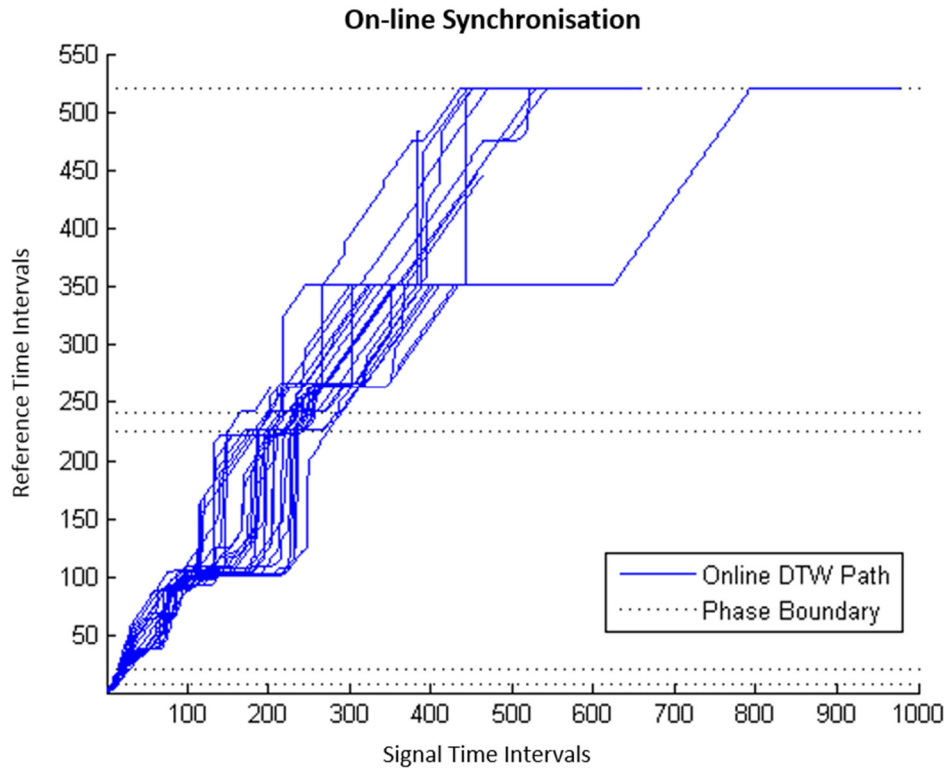


Figure 6.10: On-line DTW (γ = ∞) Synchronisation – Case Study 3

The PCCs of the on-line synchronisation and the off-line synchronisation were calculated for each batch, and the geometric means of these PCCs are shown in table 6.2. Using on-line DTW $(\gamma = \infty)$ produced the highest correlation across all the batches. The least square difference (LSD) intervals showed statistically significant differences for on-line DTW versus the RGTW methods $(\gamma = 5, 10)$ at p-value < 0.05.

Table 6.2: Global On-line Synchronisation Mean PCCs – Case Study 3

Synchronisation Method	Mean PCC	No. Samples
RGTW (γ=5)	0.9394	39
RGTW (γ=10)	0.9483	39
On-line DTW (γ= ∞)	0.9800	39

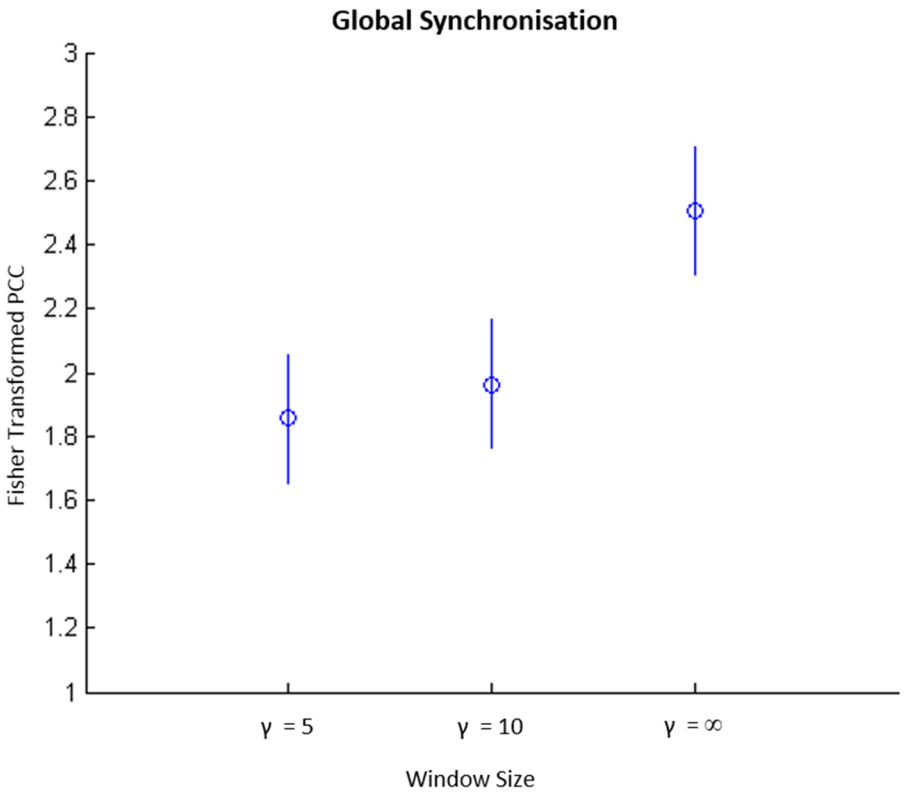


Figure 6.11: LSD Intervals for On-line Synchronisation – Case Study 3

The time taken for the analysis to be completed should also come under scrutiny. Restricting the search space by using a window greatly increases the computational efficiency of the synchronisation. These values are shown in table 6.3. On-line DTW synchronisation was only done for the full batch trajectory as opposed every time interval for RGTW, therefore the times taken for on-line DTW show the maximum amount of time taken. Near the beginning of the batch, the time taken to synchronise the trajectories would be much shorter than those shown. Nonetheless, the maximum time taken to synchronise the trajectory is still valuable information. In the FICO process, the amount of time between process variable measurements was 100s, therefore as the batch nears completion the time taken to process the data for on-line DTW becomes too great. Both RGTW algorithms were very efficient, and produced synchronised results under 1s at every point along the trajectories.

Table 6.3: Computational Efficiency of Global On-line Synchronisation – Case Study 3

	RGTW (γ=5)	RGTW (γ=10)	On-line DTW (γ= ∞)
Number of samples	20679	20679	39
Average (seconds)	0.0968	0.1769	265.92
Standard Deviation	0.0730	0.1607	102.77
Minimum	0.0015	0.0015	13.76
Maximum	0.4593	0.7266	508.59

A compromise should be sought between the time taken to process a result and the accuracy of the resulting paths. On-line DTW produced relatively accurate results, but the amount of time taken to process the results near the end of the batch was quite high. This was due to the large search space defined by the varying batch lengths. Conversely, the RGTW algorithm produced results very efficiently, but the accuracy was compromised. The window width that offers the best compromise between efficiency and accuracy lies somewhere in between.

6.5.2 Phase-wise Synchronisation

Synchronising each batch in a phase-wise manner offers some advantages over global synchronisation as it reduces the overall search space required and offers a starting point along the batch trajectory. Analysis of the geometric mean PCCs (table 6.4) and average times taken (table 6.5) show that on-line DTW is appropriate for phases 1-4, as the amount of time taken for a result to be available is significantly less than the sampling time. The increased accuracy is therefore justified by the quick processing times.

Table 6.4: Phase-wise On-line Synchronisation Geometric Mean PCCs – Case Study 3

	RGTW (γ=5)	RGTW (γ=10)	On-line DTW (γ= ∞)
Phase 1	0.9605	0.9838	0.9834
Phase 2	0.9750	0.9868	0.9876
Phase 3	0.9513	0.9632	0.9992
Phase 4	0.9349	0.9917	0.9921
Phase 5	0.6836	0.7570	0.9466

Table 6.5: Phase-wise On-line Synchronisation Mean Warping Time – Case Study 3

	RGTW (γ=5)	RGTW (γ=10)	On-line DTW (γ= ∞)
Phase 1	0.0058s	0.0069s	0.0098s
Phase 2	0.0068s	0.0089s	0.0131s
Phase 3	0.0328s	0.0492s	7.0523s
Phase 4	0.0063s	0.0078s	0.0112s
Phase 5	0.0703s	0.0720s	110.07s

In phase 5, the PCC is significantly higher for on-line DTW in compared with the PCCs for RGTW. However, the average processing time of on-line DTW is higher than the time taken between samples (the maximum time taken to process a batch was 287s – almost three times longer than the time taken between samples). One cannot reliably process the data before a new sample is available, but the increased accuracy of this technique cannot be ignored. It should also be noted that the correlation between this technique and off-line synchronisation is still not particularly high compared with the RGTW approaches, so using a window in this phase would not be appropriate if accurate warping is desired. A suggestion would be only to process the data after every fifth sample in the latter part of the phase.

A plot of the off-line and on-line paths (figure 6.12) for phase 5 show that the difference in correlation is mainly due to the identification of the final endpoint. The on-line synchronisation algorithm did not identify the reference trajectory endpoint as the endpoint of the optimal path in 12 of the 39 batches. The backwards nature of the DTW algorithm means that if a different endpoint is chosen, the entire path can be altered, which explains the differences shown here. The choice of reference trajectory in this phase was heavily influenced by the constant nature of the air flowrate variable, which may not have been the best trajectory to use for determination of the end of the phase. The temperature variable decreases well below its operating point at the end of the phase in most of the batches, but this was not well accounted for in the reference trajectory (figure 6.13) as the temperature drop falls to a value greater than its value for most of the phase. This is an important consideration for reference trajectory selection, and requires insight into the expected progress of the variables that offer the most information about the nature of the variable trajectories.

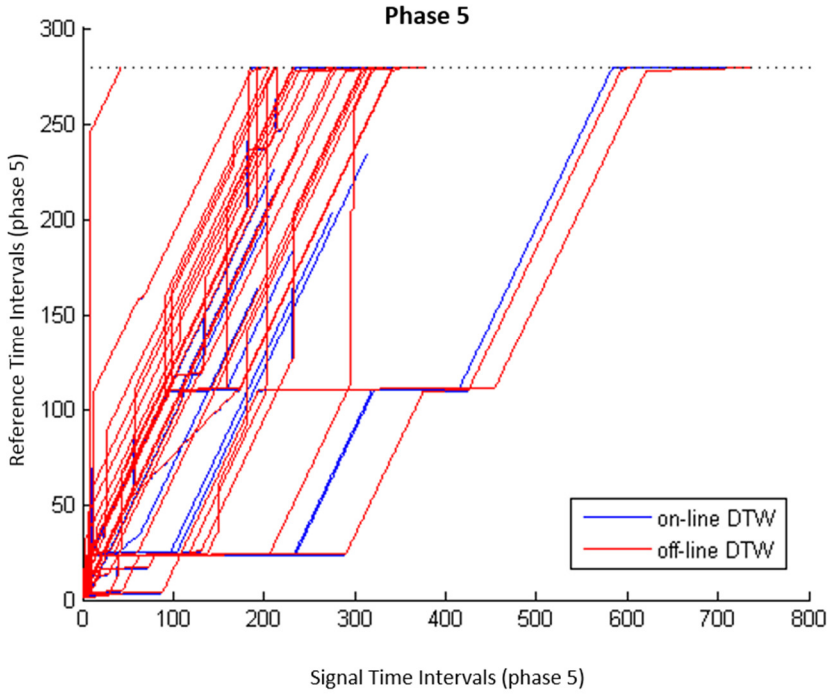


Figure 6.12: Phase 5 Synchronisation Path Comparison – Case Study 3

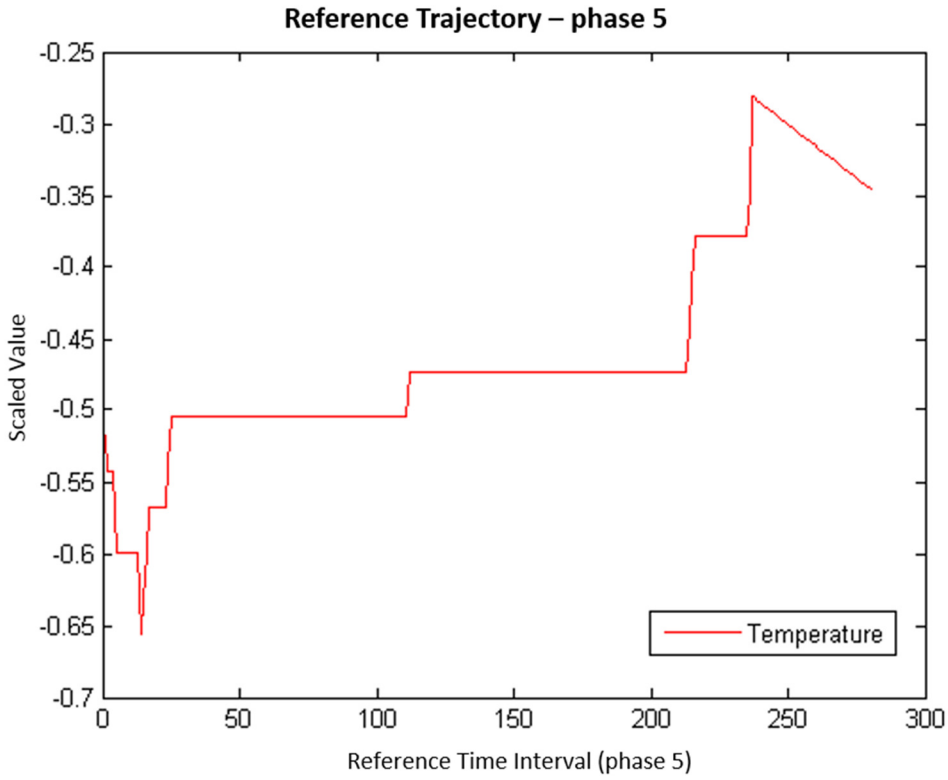


Figure 6.13: Reference Trajectory Temperature Value (phase 5) – Case study 3

Overall, synchronising the batch data on-line in a phase-wise manner is a wise approach for data such as these. The window size can be optimised for each phase, which is important when the process dynamics change across the phases. It also reduces the overall search space for each phase, which allows for a larger window width to be used for phases that require more accurate warping. However, some sort of phase identification is necessary to be able to apply the synchronisation on-line in a phase-wise manner. Expert rules were used in this approach as the phases corresponded with the different steps of the process, but the method of singular points (Doan & Srinivasan, 2008) would be able to detect phase changes. The change in gradient in the temperature and pressure variables at the end of phase 4 would be a reasonable point to indicate the start of the 5th phase.

Considering the computational speed of the RGTW approaches, global synchronisation could be used as a phase identification technique, thereafter phase-wise synchronisation could be done. The accuracy of phase identification using global synchronisation was not explicitly assessed, but the change in variable trajectories may be able to provide enough information to determine a phase change.

6.6 Summary of Results

Synchronisation of the batch trajectories in this process was of interest due to the large time variations among the batches. Significant warping of the trajectories was needed in order align the batches to a common trajectory. Global and phase-wise synchronisation were compared for the on-line approaches. The on-line DTW approach ($\gamma = \infty$) showed a statistically significant (p-value < 5) increase in warping accuracy compared with the RGTW approaches ($\gamma = 5, 10$). However, the time taken to synchronise the trajectories in on-line DTW was significant. Phase-wise synchronisation provided a way to restrict the search space for the optimal path, which allowed for timely results. On-line, phase-wise synchronisation using the DTW approach ($\gamma = \infty$) was the most appropriate synchronisation technique for this process, considering the appropriate phase identification is performed. The reference trajectory selection also played a key role in the synchronisation in phase 5, and using linear correlation between batch trajectories to select the reference trajectory might favour variables with smooth trajectories rather than the batch that represents the expected trajectory evolution, especially when highly variable data are concerned.

CHAPTER 7 Conclusions

The aim of this thesis was to apply fault detection and end-of-batch prediction to batch process data in an on-line line manner. Computationally efficient and accurate synchronisation was necessary to align the batch trajectories on-line, and phase division based on correlation structure was also included in an attempt to improve the accuracy of the fault detection and end-of-batch quality predictions. On-line identification of the correct phase was necessary to apply the correct model.

In this chapter, general conclusions are made about synchronisation, phase division and on-line fault detection and end-of-batch quality prediction. Thereafter, the objectives from chapter 1 are revisited and the fulfilment thereof is discussed.

7.1.1 Synchronisation

The Relaxed Greedy Time Warping (RGTW) algorithm was implemented to synchronise incoming batch trajectories in an accurate and timely manner such that potential faults could be identified and end-of batch-predictions could be obtained in near-real time. The on-line synchronization accuracy was compared to the alignment obtained from pre-processing the same data off-line using conventional off-line dynamic time warping. The metric used to compare was the Pearson's Correlation Coefficient (PCC).

An attractive feature of the RGTW algorithm is that the window width could be adjusted to increase the search space for the synchronisation path, which added flexibility to the approach. The window size could be adjusted to a value larger than the number of time intervals in the largest batch to be synchronised, which was similar to the on-line DTW approach, but with the limits set by the paths of the training data.

On-line RGTW was computationally efficient, and was able to produce results very quickly. Even using a window width of $\gamma = 50$, synchronisation could be performed in less than a couple of seconds. This is a major advantage of the approach compared with off-line DTW, which is computationally expensive. It also allows for extensions to be made to the platform that could improve accuracy, such as Hybrid Dynamic Derivative Time Warping (HDDTW).

The greedy nature of the RGTW algorithm revealed suboptimal warping in some areas of the process that had a material effect on the results obtained for fault detection and end-of-batch quality prediction. Warping differences were apparent in parts of the batch trajectories where the trajectories were flatter. The warping accuracy in these regions generally improved as the window size was increased.

The warping of the time axis caused information critical to the end-of-batch quality to be lost in the first case study. This is because the end-of-batch quality was directly dependent on time, and warping distorted this information. A smaller window did not affect this error as much. Including time as a process variable could help retain this information.

The comparison of off-line DTW with the Indicator Variable approach showed differences in the trajectory alignment. The DTW algorithm unnecessarily stretched and compressed the paths at certain points, whereas the IV preferred a constant, linear path. In the penicillin cultivation case study, the IV was assumed to an accurate representation of the progress of the batches. The RGTW approaches showed better linear correlation with the IV approach than with the DTW approach for most of the window widths investigated. Off-line DTW synchronisation therefore may not offer the most accurate synchronisation for such trajectories, which would affect the models that are trained on these data.

Synchronising trajectories on-line per phase can help increase synchronisation accuracy, although appropriate phase identification is necessary to apply the phase-specific synchronisation parameters.

7.1.2 Phase Division

The effect of partitioning the data based on operational and correlation phases was explored. The MPPLS algorithm was implemented for batch-wise unfolding as a way of automatically identifying phase divisions based on correlation structure. Especially in batch-wise unfolding with future measurements imputed as zeros, splitting the data into separate models over the course of a batch trajectory could improve performance, but due to the number of missing data that comprised the model, a reduction in prediction accuracy at certain points of the process could be observed. Phase splits should not be made during critical-to-quality parts of the process, as prediction power will be lost.

The MPPLS algorithm tries to address this by specifically searching for the optimal split of data that will result in the best prediction accuracy. However, the algorithm still uses greedy searches to identify the partitions, and sub-optimal partitioning may occur. This was shown in the second application of the MPPLS algorithm. The data were split into 6 correlation phases; many more partitions than there were operational phases. The model showed statistically significant reductions in the MSE and the conclusion could be drawn that the MP algorithm was able to identify data splits based on the reduction of prediction error, and that these partitions would result in improved prediction power of the models.

In on-line applications, there was evidence that the MPPLS identified phase divisions provided comparable or better performance for end-of-batch quality prediction. Predictions were closer to their true values for most of the batch trajectory in the second case study compared with the other model sets. The identification in changing linear correlation structure greatly aided the ability of the models to make accurate predictions.

7.1.3 On-line Fault Detection and End-of-Batch Quality Prediction

The existing batch process monitoring platform was successfully modified to be able to process data in an on-line fashion and generate statistics used for fault identification and to predict the end-of-batch quality. Improper synchronisation was found to have an effect on the monitoring statistics generated and influenced the fault detection. Poorly synchronised parts of the batch trajectory resulted in a high false alarm rate during that part of the process.

Overall, the platform was able to provide near real-time results that could be used on-line for monitoring and end-of-batch quality prediction once the limits have been adjusted for normal operating conditions.

7.1.4 Fulfilment of Objectives

Objective	Fulfilment
Determination of whether on-line synchronisation could produce computationally efficient results comparable to off-line synchronisation results	Results of on-line synchronisation were compared to off-line synchronisation results. It was found that comparable results could be obtained, but there were some inaccuracies when the variable trajectories were flatter. The necessity to synchronise the trajectories globally as opposed to in a phase-wise manner on-line also affected the accuracy.
Determination of whether batch data could automatically be partitioned based on correlation structure	The identification of correlation phases was achieved successfully using the MPPLS algorithm, which showed a statistically significant reduction in MSE for quality prediction in one case study.
Determination of whether the on-line monitoring platform could provide accurate but timely monitoring and end-of-batch quality prediction results	The platform was able to provide timely results, but the false alarm rate was high in certain regions due to improper synchronisation. Faults were generally detected accurately, but time warping affected the identification of faults when the processing time was a quality variable
Comparison of on-line monitoring and end-of-batch quality prediction performance of models trained on batch data partitioned based on correlation structure to conventional modelling procedures	Models built on the three different phase division approaches were compared. Superior end-of-batch quality prediction was achieved by partitioning data according to correlation phases in one case study, otherwise performance was comparable to the conventional approaches.

CHAPTER 8 Recommendations

The conclusions given in the previous chapter give some insight into the ability of the platform to detect faults and provide end-of-batch quality predictions, as well of the accuracy of the on-line synchronisation and phase division. This chapter provides some recommendations to improve the overall platform.

8.1 Synchronisation

On-line synchronisation using RGTW was computationally efficient, but suffered from poor accuracy at certain points in the process. Inaccurate synchronisation of flat trajectories has been reported on when the Euclidean distance is used as a metric, and this part of synchronisation could be improved by incorporating a different metric. Hybrid Derivative Dynamic Time Warping (HDDTW) offers a solution for accurate warping in these regions with flatter trajectories. A critique of the approach was that the computational time of this method was increased because of the extra derivatives that had to be calculated along with the cumulated distances. This is not too much of a problem if this approach is incorporated with RGTW because of computational efficiency of the algorithm.

Variation of the window size across the batch trajectory could enhance the synchronisation accuracy. During flatter parts of the batch trajectories, larger window widths could be used to account for the slower dynamics. In regions where the trajectories exhibit much more variation, smaller window widths can be used. The reference trajectory indexes can be used as a way to identify when a window size adjustment should be made – once the algorithm has synchronised to a certain reference point, the window size could change.

On-line synchronisation per phase should be integrated into the platform. The slab scaling parameters used affect the warping accuracy, and slab scaling per phase would probably improve the accuracy of the warping. Some sort of phase identification would be necessary, but using synchronisation to the reference trajectory could provide an initial basis for phase identification. Incorporating singular points (SPs) could be a way of improving phase identification.

8.2 Phase Division

The MPPLS algorithm performed well in terms of end-of-batch predictions, and fault identification was quite accurate, or at least comparable to the global and operational phase models. Extension of this algorithm to run the algorithm at different initial parameters and automatically compare the different model sets would further increase the automation of the approach.

References

Arteaga, F., & Ferrer, A. (2002). Dealing with missing data in MSPC: Several methods, different interpretations, some examples. *Journal of Chemometrics*, *16*(8–10), 408–418.

Barnard, J. P., 2015. *Software Notes - BPM Toolbox v1,* Stellenbosch: Centre for Process Monitoring.

Beckner, M., Ivey, M. L. & Phister, T. G., (2011). Microbial contamination of fuel ethanol fermentations. *Letters in Applied Microbiology,* 53(4), pp. 387-394.

Birol, G., Ündey, C., & Çinar, A. (2002). A modular simulation package for fed-batch fermentation: Penicillin production. *Computers and Chemical Engineering*, *26*(11), 1553–1565.

Bro, R. (1996). Multiway calibration. Multilinear PLS. *Journal of Chemometrics*, *10*(1), 47–61.

Bro, R. (1997). PARAFAC. Tutorial and applications. *Chemometrics and Intelligent Laboratory Systems*, *38*(2), 149–171.

Brown, M., & Rabiner, L. (1982). Dynamic time warping for isolated word recognition based on ordered graph searching techniques. *ICASSP '82. IEEE International Conference on Acoustics, Speech, and Signal Processing*, *7*, 1255–1258.

Camacho, J., & Picó, J. (2006). Multi-phase principal component analysis for batch processes modelling. *Chemometrics and Intelligent Laboratory Systems*, *81*(2), 127–136.

Camacho, J. P., Picó, J. & Ferrer, A., (2007). *New Methods Based on the Projection to Latent Structures for Monitoring, Prediction and Optimization of Batch Processes [PhD Dissertation]*. Technical University of Valencia.

Camacho, J., Picó, J., & Ferrer, A. (2008). Multi-phase analysis framework for handling batch process data. *Journal of Chemometrics*, *22*(11–12), 632–643.

Chalupa, P., Novák, J., & Bobál, V. (2011). Detailed Simulink Model of Real Time Three Tank System. *Recent Researches in Circuits, Systems, Communications and Computers*, 161–166.

Chandel, A. K. et al., (2007). Economics and environmental impact of bioethanol production technologies: an appraisal. *Biotechnology and Molecular Biology Review,* 2(1), pp. 14-32.

Chen, J. & Liu, K., (2003). On-line batch process monitoring using dynamic pca and dynamic pls models. *Chemical Engineering Science,* pp. 93-109.

Crundwell, F. K., Moats, M. S., Ramachandran, V., Robinson, T. G., Davenport, W. G., Crundwell, F. K., … Davenport, W. G. (2011). Chapter 37 – Refining of the Platinum-Group Metals. In *Extractive Metallurgy of Nickel, Cobalt and Platinum Group Metals* (pp. 489–534).

Doan, X.-T., & Srinivasan, R. (2008). Online monitoring of multi-phase batch processes using phase-based multivariate statistical process control. *Computers & Chemical Engineering*, *32*(1–2), 230–243.

Dunteman, G. H. (1989). *Principal Components Analysis*. Newbury Park, CA: SAGE Publications, Inc.

Fransson, M., & Folestad, S. (2006). Real-time alignment of batch process data using COW for on-line process monitoring. *Chemometrics and Intelligent Laboratory Systems*, *84*(1–2 SPEC. ISS.), 56–61.

Frank, P. M. (1990). Fault Diagnosis in Dynamic Systems Usind Analytical Knowledge-based Redundancy - A Survey and Some New Results. *Automatica,* 26(3), pp. 459-474.

Garcia-Alvarez, D., Fuente, M. J., & Sainz, G. I. (2012). Fault detection and isolation in transient states using principal component analysis. *Journal of Process Control*, *22*(3), 551–563.

Ge, Z., & Song, Z. (2014). Online Monitoring and Quality Prediction of Multiphase Batch Processes with Uneven Length Problem.

Gins, G., Van Den Kerkhof, P., & Van Impe, J. F. M. (2012). Hybrid derivative dynamic time warping for online industrial batch-end quality estimation. *Industrial and Engineering Chemistry Research*, *51*(17), 6071–6084.

González-Martínez, J. M., Ferrer, A., & Westerhuis, J. A. (2011). Real-time synchronization of batch trajectories for on-line multivariate statistical process control using Dynamic Time Warping. *Chemometrics and Intelligent Laboratory Systems*, *105*(2), 195–206.

Höskuldsson, A. (1988). PLS Regression Methods. *Journal of Chemometrics,* Volume 2, pp. 211-228.

Itakura, F. (1975). Minimum Prediction Residual Principle Applied to Speech Recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing,* pp. 67-72.

Kanagasabai, N., & Jaya, N. (2014). Design of Multiloop Controller for Three Tank Process Using Cdm Techniques. *International Journal on Soft Computing*, *5*(2), 11–20.

Kassidas, A. (1998). *Fault Detection and Diagnosis in Dynamic Multivariable Chemical Processes Using Speech Recognition Methods*. McMaster University. Retrieved from http://journals.cambridge.org/abstract_S0165115300023299

Kassidas, A., Macgregor, J. F., & Taylor, P. A. (1998). Synchronization of batch trajectories using dynamic time warping. *AIChE Journal*, *44*(4), 864–875.

Kourti, T. (2002a). Multivariate dynamic data modeling for analysis and statistical process control of batch processes, start-ups and grade transitions. *Journal of Chemometrics,* Volume 17, pp. 93-109.

Kourti, T. (2002b). Process Analysis and Abnormal Situation Detection: From Theory to Practice. *IEEE Control Systems Magazine,* pp. 10-25.

Kourti, T. (2003a). Abnormal situation detection, three-way data and projection methods; robust data archiving and modeling for industrial applications. *Annual Reviews in Control*, *27*(2), 131–139.

Kourti, T. (2003b). Multivariate dynamic data modeling for analysis and statistical process control of batch processes, start-ups and grade transitions. *Journal of Chemometrics*, *17*(1), 93–109.

Kourti, T. (2006). Process Analytical Technology Beyond Real-Time Analyzers: The Role of Multivariate Analysis. *Critical Reviews in Analytical Chemistry,* Volume 36, pp. 257-278.

Kourti, T., Lee, J., & MacGregor, J. F. J. F. (1996). Experiences with industrial applications of projection methods for multivariate statistical process control. *Computers & Chemical Engineering*, *20*(96), S745–S750.

Kroonenberg, P. M. (1992). Three-mode Component Models A Survey of the Literature. *Statistica Applicata,* 4(4), pp. 619-633.

Lakshminarayanan, S., Gudi, R., Shah, S. & Nandakumar, K. (1996). *Monitoring batch processes using multivariate statistical tools: extensions and practical issues*. SAn Francisco, s.n., pp. 241-246.

Lee, J. M., Yoo, C. K., & Lee, I. B. (2004). Enhanced process monitoring of fed-batch penicillin cultivation using time-varying and multivariate statistical analysis. *Journal of Biotechnology*, *110*(2), 119–136.

Lu, N., & Gao, F. (2005). Stage-Based Process Analysis and Quality Prediction for Batch Processes. *Industrial & Engineering Chemistry Research*, *44*(10), 3547–3555.

Lu, N., Gao, F., & Wang, F. (2004). Sub-PCA Modeling and on-line Monitoring Strategy for Batch Processes. *AIChE Journal*, *50*(1), 255–259.

Lu, N., Gao, F., Yang, Y., & Wang, F. (2004). PCA-based modeling and on-line monitoring strategy for uneven-length batch processes. *Industrial & Engineering Chemistry Research*, *43*, 3343–3352.

Moita, R. D., Gomes, V. M., Saravia, P. M. & Reis, M. S. (2014). An Extended Comparitive Study of Two- and Three-Way Methodologies for the On-line Moniotoring of Batch Processes. *Computer Aided Chemical Engineering,* Volume 33, pp. 517-522.

Nielsen, N.-P. V., Carstensen, J. M. & Smedsgaard, J. (1998). Aligning of single and multiple wavelength chromatographic profiles form chemometric data analysis using correlation. *Journal of Chromatography,* Volume A 805, pp. 17-35.

Nomikos, P. & MacGregor, J. F., (1994). Monitoring batch processes using principal component analysis. *AIChe Journal,* 40(8), pp. 1361-1375.

Nomikos, P., & Macgregor, J. F. (1995). Multivariate SPC Charts for Batch Monitoring Processes. *Technometrics*, *37*(1), 41–59.

Nomikos, P., & MacGregor, J. F. (1995). Multi-way partial least squares in monitoring batch processes. *Chemometrics and Intelligent Laboratory Systems*, *30*(1), 97–108.

Pravdova, V., Walczak, B. & Massart, D. L. (2002). A comparison of two algorithms for warping of analytical signals. *Analytica Chimica Acta,* Volume 456, pp. 77-92.

Ramaker, H. J., Van Sprang, E. N. M., Westerhuis, J. A., & Smilde, A. K. (2003). Dynamic time warping of spectroscopic BATCH data. *Analytica Chimica Acta*, *498*(1–2), 133–153.

Ramaker, H., van Sprang, E. N., Westerhuis, J. A. & Smilde, A. K., (2005). Fault detection properties of global, local and time evolving models for batch process monitoring. *Journal of Process Control,* pp. 799-805.

Ratanamahatana, C. A., & Keogh, E. (2004). Making Time-series Classification More Accurate Using Learned Constraints. In *SIAM International Conference on Data Mining* (pp. 11–22).

Ratanamahatana, C. A., & Keogh, E. (2005). Three Myths about Dynamic Time Warping. In *SIAM International Conference on Data Mining* (pp. 506–510). Newport Beach, CA.

Rato, T. J., Rendall, R., Gomes, V., Chin, S. T., Chiang, L. H., Saraiva, P. M., & Reis, M. S. (2016). A Systematic Methodology for Comparing Batch Process Monitoring Methods: Part I-Assessing Detection Strength. *Industrial and Engineering Chemistry Research*, *55*(18), 5342–5358.

Sagués, F. & Epstein, I. R., (2003). Nonlinear chemical dynamics. *Dalton Transactions,* pp. 1201-1217.

Sakoe, H. & Chiba, S., (1978). Dynamic Proramming Algorithm Optimization for Spoken Word Recognition. *IEEE Transactions on Acoustic, Speech and Signal Process,* pp. 43-49.

Shen, F., Ge, Z., & Song, Z. (2015). Multivariate Trajectory-Based Local Monitoring Method for Multiphase Batch Processes. *Industrial & Engineering Chemistry Research*, *54*(4), 1313–1325.

Sobhani, M. H. (2011). Performance Assessment of the Designed Controllers for Three-Tank Benchmark System. *International Journal of Instrumentation and Control Systems*, *1*(2), 1–11.

Srinivasan, R., & Qian, M. S. (2005). Off-line temporal signal comparison using singular points augmented time warping. *Industrial and Engineering Chemistry Research*, *44*(13), 4697–4716.

Tomasi, G., Van Den Berg, F., & Andersson, C. (2004). Correlation optimized warping and dynamic time warping as preprocessing methods for chromatographic data. *Journal of Chemometrics*, *18*(5), 231–241.

Ündey, C. & Çinar, A., (2002). Statistical Monitoring of Multistage, Multiphase Batch Processes. *IEEE Control Systems Magazine,* pp. 40-52.

Ündey, C., Ertunç, S., Çınar, A., Undey, C., Ertunc, S., & Cinar, A. (2003). Online Batch/Fed-Batch Process Performance Monitoring, Quality Prediction, and Variable-Contribution Analysis for Diagnosis. *Industrial & Engineering Chemistry Research*, *42*, 4645–4658.

Ündey, C., Tatara, E., & Çinar, A. (2004). Intelligent real-time performance monitoring and quality prediction for batch/fed-batch cultivations. *Journal of Biotechnology*, *108*(1), 61–77.

Wan, J., Marjanovic, O., & Lennox, B. (2014). Uneven batch data alignment with application to the control of batch end-product quality. *ISA Transactions*, *53*(2), 584–590.

Westad, F., Gidskehaug, L., Swarbrick, B., & Flåten, G. R. (2015). Assumption free modeling and monitoring of batch processes. *Chemometrics and Intelligent Laboratory Systems*, *149*, 66–72.

Westerhuis, J. A, Kourti, T., & Macgregor, J. F. (1999). Comparing alternative approaches for multivariate statistical analysis of batch process data. *Journal of Chemometrics*, *13*(3–4), 397–413.

Wold, H., (1974). Casual flows with latent variables: Partings of the ways in the light of NIPALS modelling. *European Economic Review,* 5(1), pp. 67-86.

Wold, S., Kettach, N., Friden, H. & Holmberg, A., (1998). Modelling and diagnostics of batch processes and analogous kinetic experiments. *Chemometrics and Intelligent Laboratory Systems,* Volume 44, pp. 331-340.

Wisner, J. & Stanley, L., (2008). *Process Management: Creating Value Along the Supply Chain.* 1st ed. Mason: LEAP Publishing Services, Inc..

Yao, Y., & Gao, F. (2009). A survey on multistage/multiphase statistical modeling methods for batch processes. *Annual Reviews in Control*, *33*(2), 172–183.

Yzelle, C., (2013). *Batch Process Monitoring Quality Prediction,* Stellenbosch: Centre for Process Monitoring.

APPENDIX A: Case Study Initial Conditions

Table A.1: Variable Conditions for Three-tank Simulation

Batch No.	Q_1 flow rate (m^3/s)	Q_1 flow rate (m^3/s)	Pump 1 Decrease Time (s)	Pump 2 Decrease Time (s)
1	1.15×10^{-4}	1.15×10^{-4}	34	57
2	1.21×10^{-4}	1.16×10^{-4}	35	60
3	1.15×10^{-4}	1.15×10^{-4}	35	56
4	1.15×10^{-4}	1.22×10^{-4}	34	63
5	1.15×10^{-4}	1.21×10^{-4}	33	58
6	1.15×10^{-4}	1.22×10^{-4}	28	57
7	1.21×10^{-4}	1.21×10^{-4}	31	64
8	1.22×10^{-4}	1.21×10^{-4}	31	64
9	1.21×10^{-4}	1.22×10^{-4}	26	59
10	1.15×10^{-4}	1.21×10^{-4}	27	57
11	1.15×10^{-4}	1.15×10^{-4}	34	64
12	1.15×10^{-4}	1.15×10^{-4}	34	58
13	1.21×10^{-4}	1.21×10^{-4}	29	63
14	1.15×10^{-4}	1.15×10^{-4}	26	56
15	1.22×10^{-4}	1.22×10^{-4}	26	61
16	1.15×10^{-4}	1.21×10^{-4}	25	63
17	1.15×10^{-4}	1.21×10^{-4}	32	61
18	1.15×10^{-4}	1.15×10^{-4}	28	60
19	1.22×10^{-4}	1.21×10^{-4}	27	62
20	1.21×10^{-4}	1.15×10^{-4}	26	63
21	1.22×10^{-4}	1.22×10^{-4}	29	60

Table A.2: Initial Conditions for Penicillin Cultivation Process Cultivation

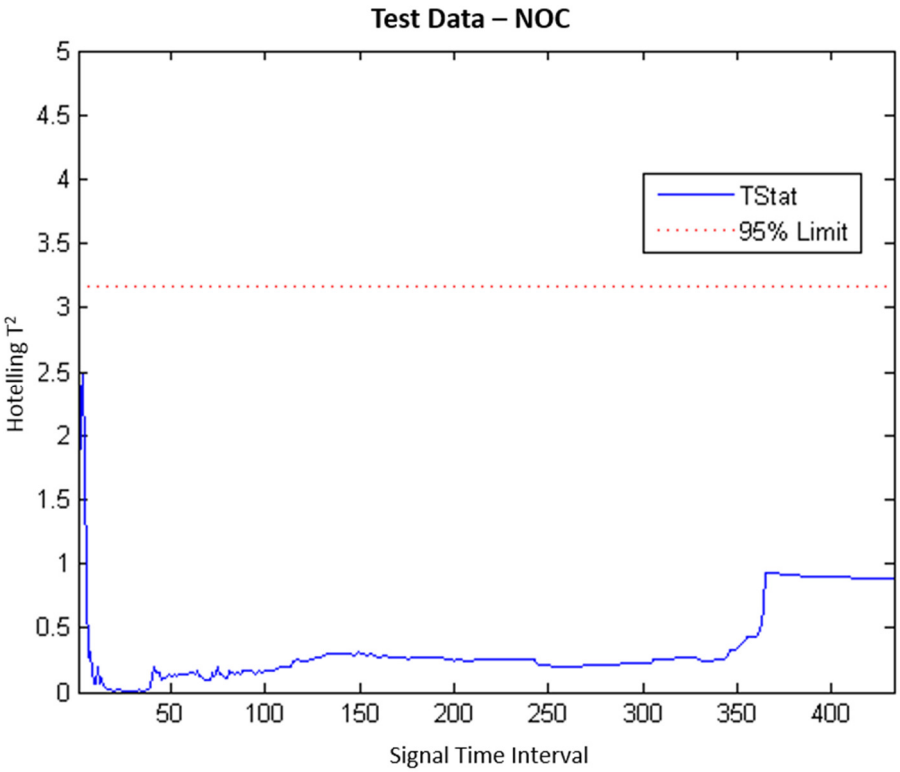
Batch no.	Glucose Conc. (g/L)	dO_2 Conc. (mmole/L)	Culture Vol. (L)	CO_2 Conc. (mmole/L)	pH	Fermenter Temp. (K)
1	17.50	1.08	102.12	0.59	4.86	297.78
2	17.33	1.07	103.00	0.57	4.89	298.06
3	15.64	1.05	101.96	0.55	5.21	298.94
4	14.78	1.20	101.08	0.57	4.55	297.73
5	17.21	1.13	102.13	0.54	5.01	297.88
6	17.89	1.05	102.84	0.82	4.53	297.81
7	15.87	1.00	101.22	0.60	5.07	297.08
8	15.31	1.06	101.52	0.99	4.87	298.84
9	16.57	1.06	101.66	0.62	5.35	297.04
10	14.47	1.04	103.24	0.51	4.72	298.18
11	15.80	1.09	103.58	0.82	4.96	297.97
12	15.67	1.04	103.72	0.87	5.24	298.02
13	15.91	1.01	102.69	0.73	4.55	298.93
14	15.60	1.03	102.50	0.79	4.51	298.21
15	18.00	1.02	101.15	0.70	4.52	297.07
16	17.60	1.15	102.11	0.57	5.19	297.68
17	14.70	1.04	101.09	0.54	5.08	298.51
18	15.35	1.12	103.76	0.99	4.65	298.05
19	16.62	1.07	100.78	0.95	4.61	297.03
20	17.89	1.09	102.62	0.82	5.17	298.60
21	14.71	1.07	101.44	0.64	4.60	298.25
22	16.85	1.14	100.01	0.71	4.63	297.05
23	14.24	1.08	103.94	0.87	5.06	298.92
24	15.00	1.11	100.20	0.54	4.82	297.10

25	16.01	1.08	103.83	0.95	5.18	298.13
26	14.75	1.07	103.83	0.88	4.60	297.00
27	17.02	1.20	102.77	0.69	4.69	298.13
28	16.62	1.17	100.39	0.69	5.33	297.03
29	15.69	1.13	102.24	0.70	4.62	298.90
30	16.27	1.16	102.38	0.69	4.78	297.45
31	15.50	1.12	102.35	0.73	4.89	297.95
32	15.48	1.20	103.89	0.53	4.96	298.45
33	15.79	1.12	102.10	0.98	4.75	298.72
34	14.40	1.07	103.73	0.80	4.53	298.77
35	17.98	1.13	103.36	0.66	4.96	298.82
36	14.15	1.09	103.61	0.88	4.72	298.58
37	17.50	1.04	100.98	0.84	4.62	298.55
38	17.12	1.12	103.23	0.99	4.53	297.90
39	14.60	1.12	102.15	0.93	5.19	297.20
40	14.26	1.08	100.57	0.51	4.87	298.35
41	17.84	1.01	101.70	0.67	5.01	297.50
42	15.40	1.06	100.51	0.80	5.35	298.33
43	17.01	1.01	101.88	0.64	5.13	297.96
44	16.33	1.17	100.76	0.75	5.25	297.45
45	17.38	1.04	101.15	0.77	4.98	297.75
46	17.49	1.10	103.29	0.54	5.31	297.34
47	15.15	1.18	103.91	0.55	4.74	298.58
48	15.06	1.11	100.15	0.64	5.01	297.86
49	17.78	1.16	103.98	0.80	4.53	297.20
50	15.34	1.12	103.76	0.61	5.01	297.65

51	15.63	1.20	103.31	0.51	4.87	298.34
52	15.11	1.11	101.28	0.77	5.11	297.11
53	17.89	1.04	102.67	0.98	4.93	297.34
54	16.15	1.08	102.92	0.59	4.54	297.37
55	15.16	1.07	101.75	0.95	5.08	298.09
56	14.76	1.07	100.38	0.77	4.73	297.78
57	16.83	1.12	103.04	0.87	4.54	298.01
58	17.47	1.02	103.16	0.94	4.72	297.00
59	14.88	1.04	100.59	0.51	4.94	298.61
60	17.84	1.06	101.68	0.62	4.80	297.78

APPENDIX B: Three Tank Simulation Monitoring Charts

a)



b)

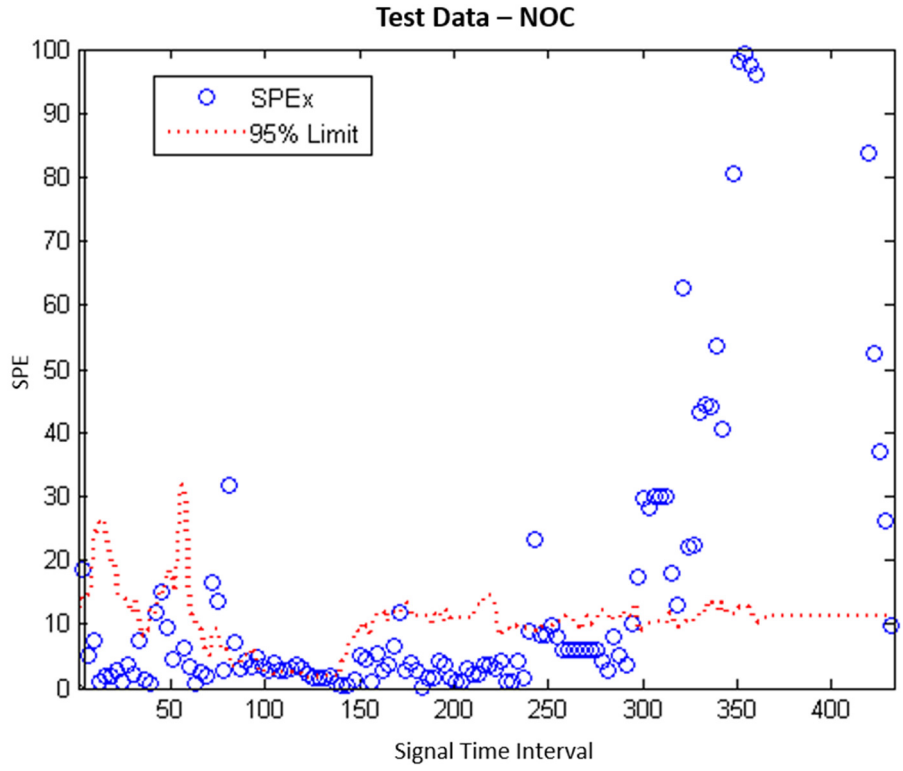
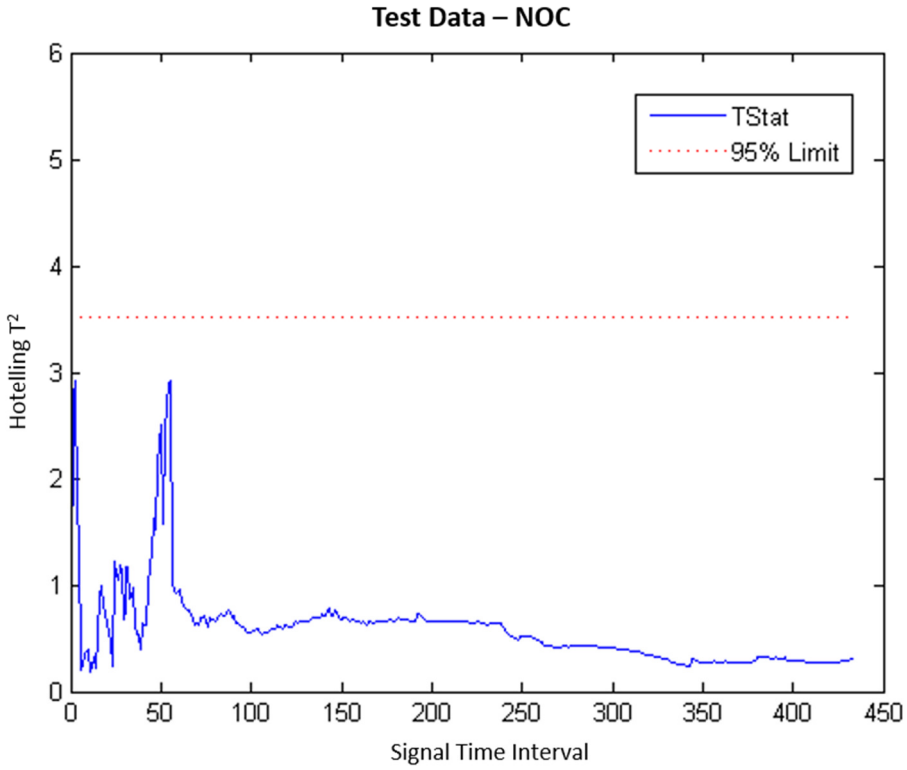


Figure B.1: NOC Test Data Monitoring Charts (Global Model, γ = 5) – Case Study 1

a)



b)

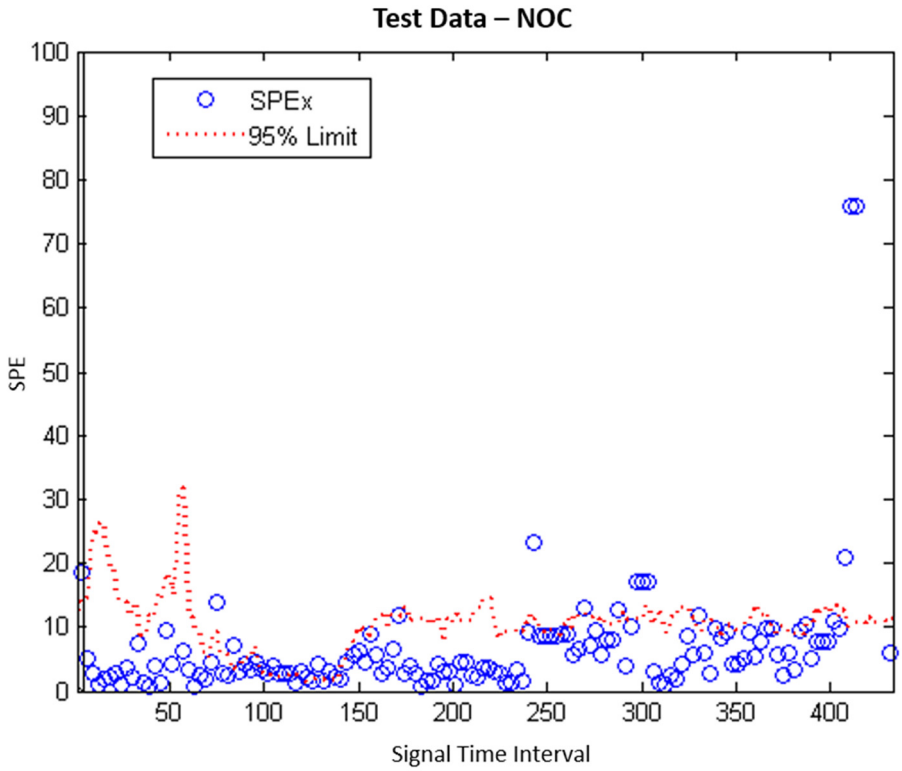
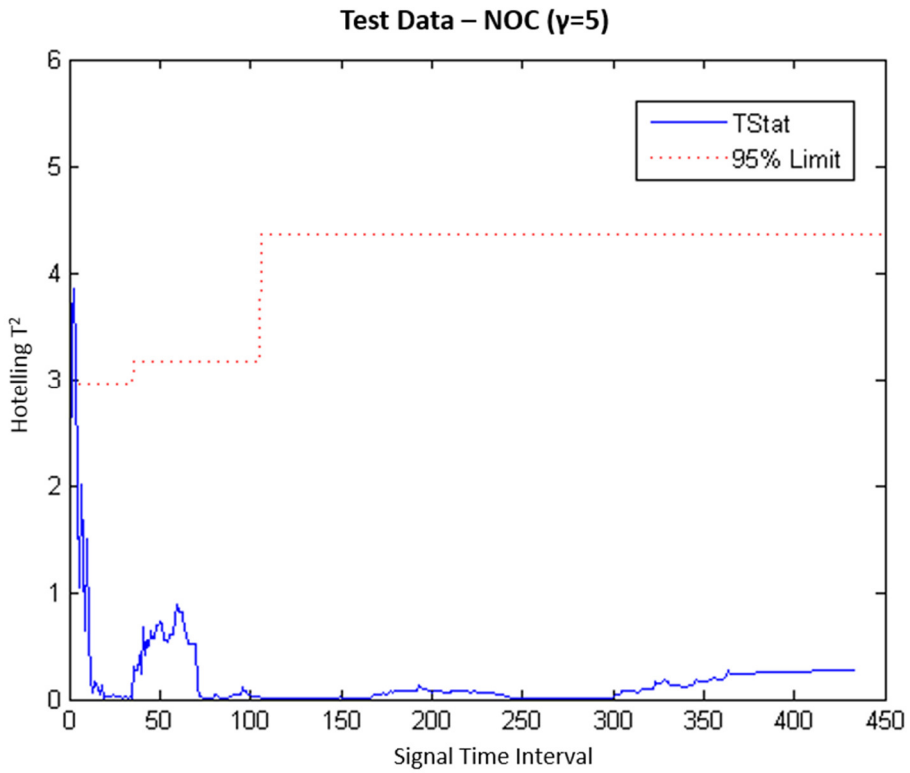


Figure B.2: NOC Test Data Monitoring Charts (Global Model, γ = 20) – Case Study 1

a)



b)

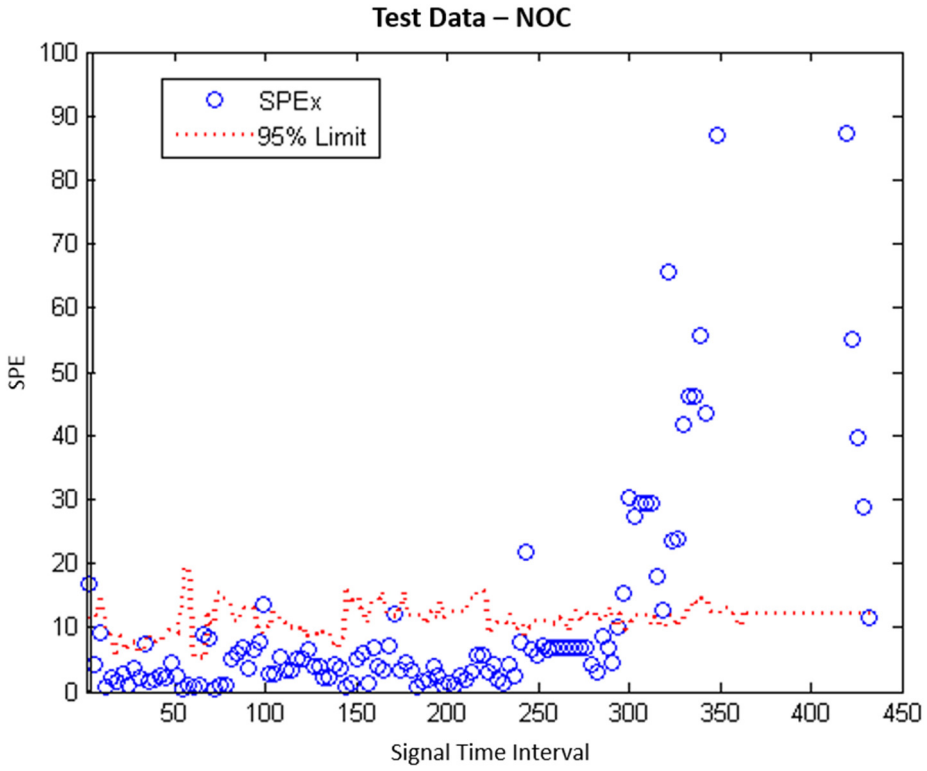
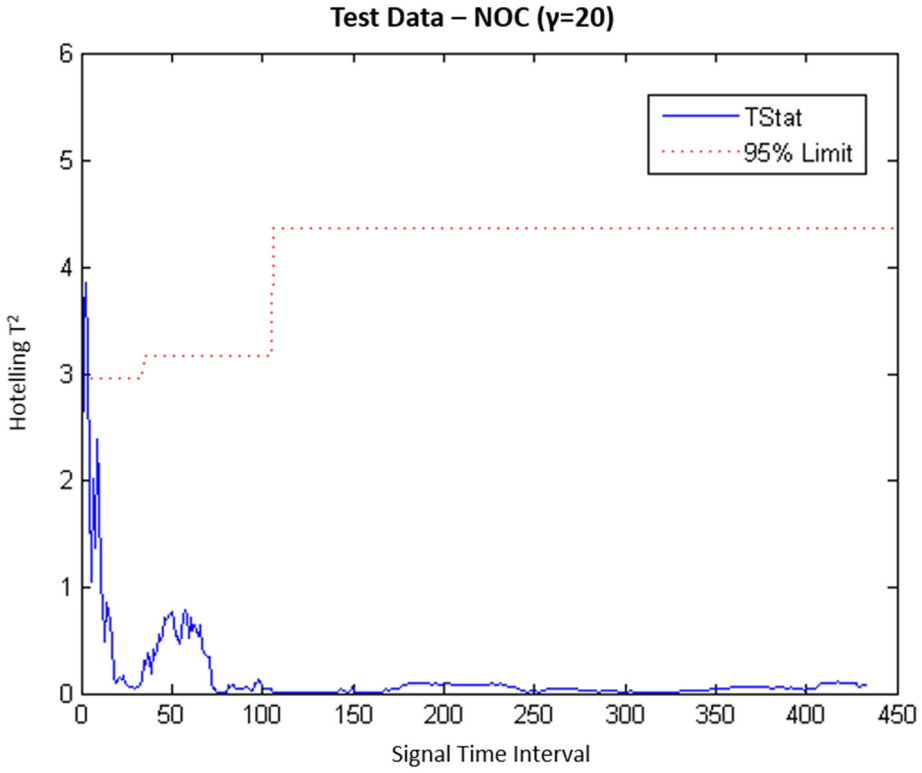


Figure B.3: NOC Test Data Monitoring Charts (Operational Phase Models, γ = 5) – Case Study 1

a)



b)

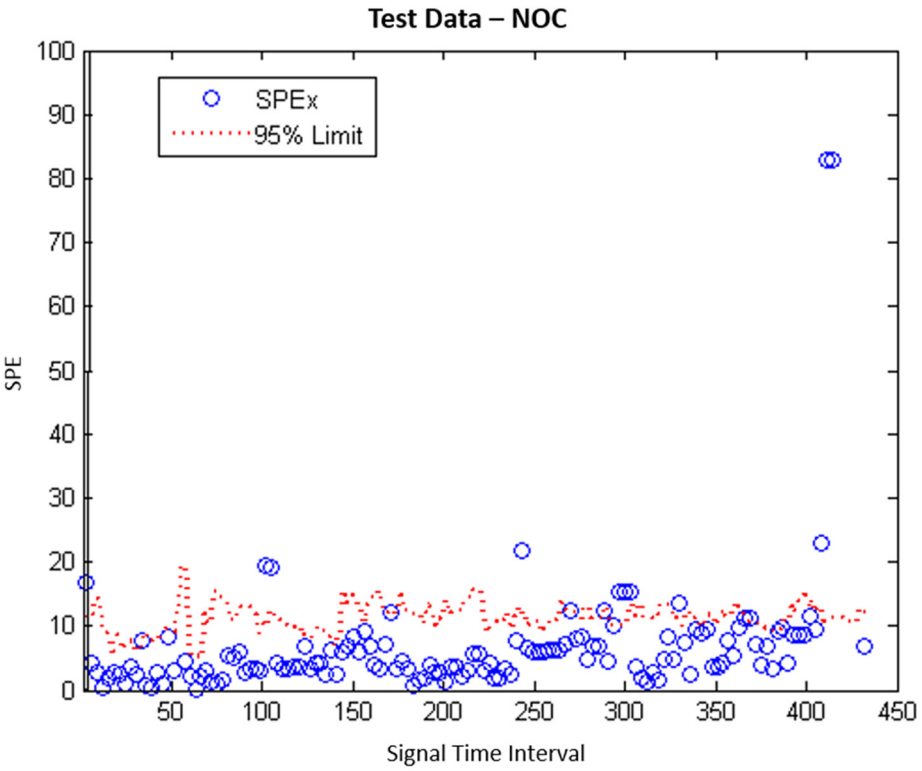
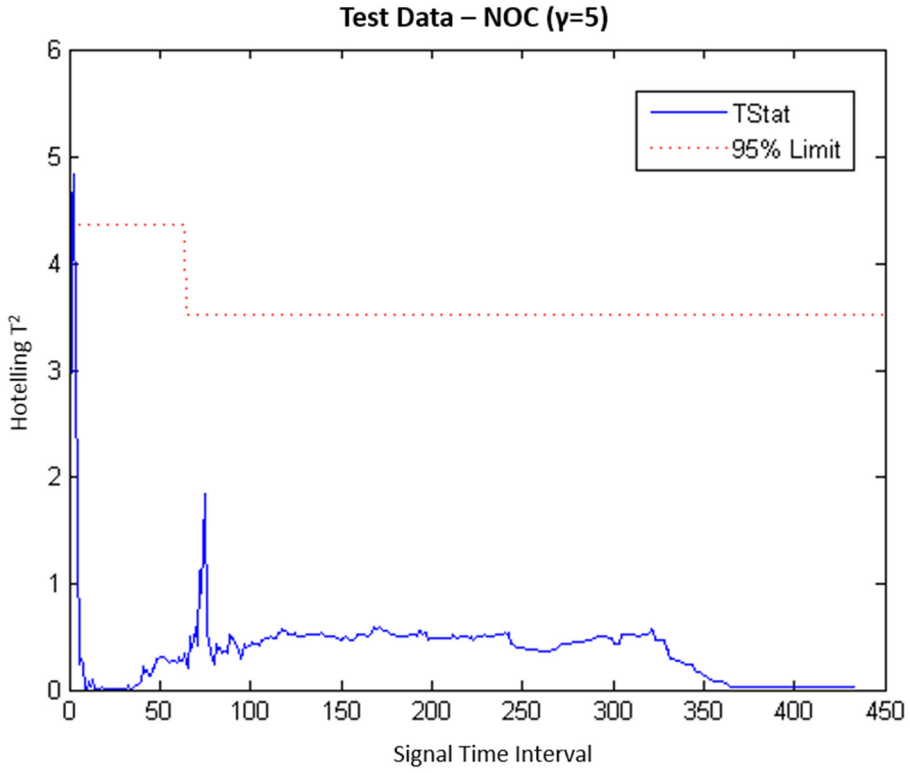


Figure B.4: NOC Test Data Monitoring Charts (Operational Phase Models, γ = 20) – Case Study 1

a)



b)

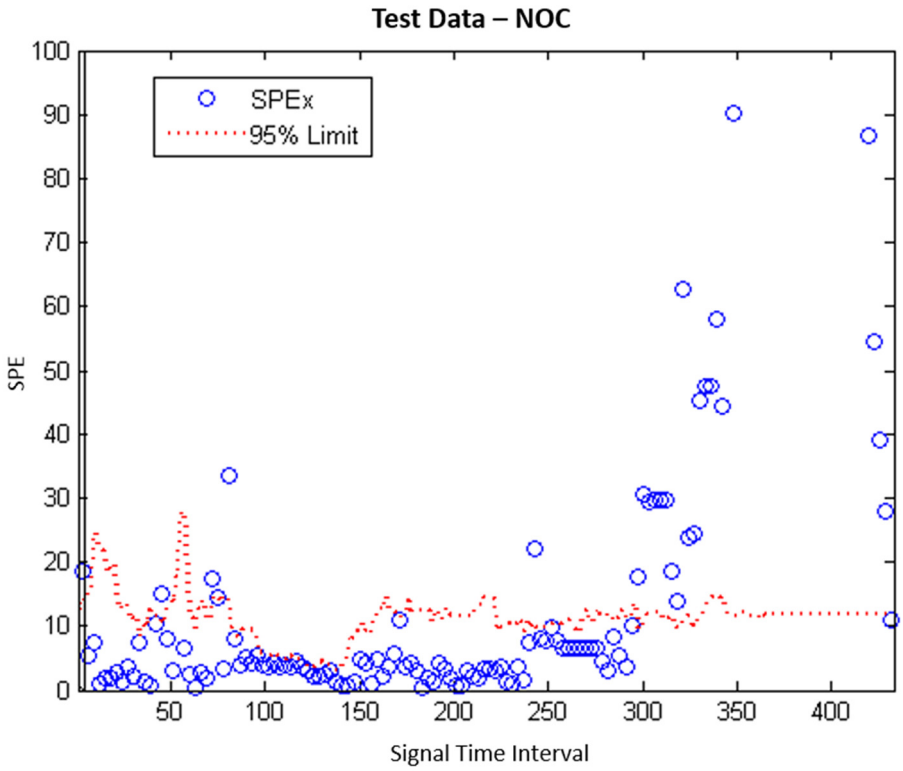
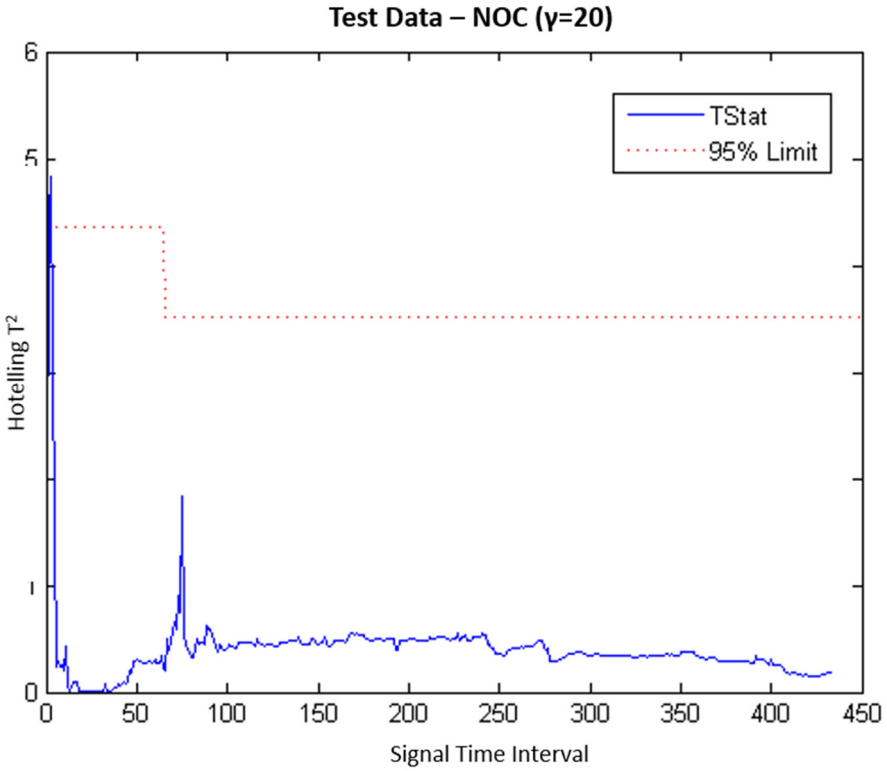


Figure B.5: NOC Test Data Monitoring Charts (Multiphase Models, γ = 5) – Case Study 1

a)



b)

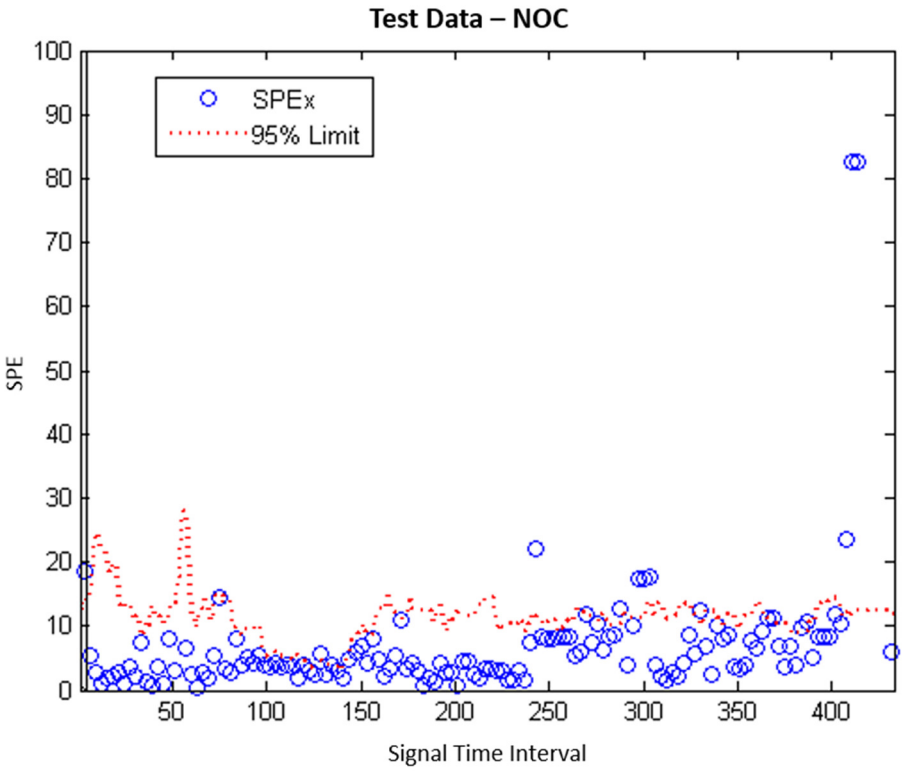
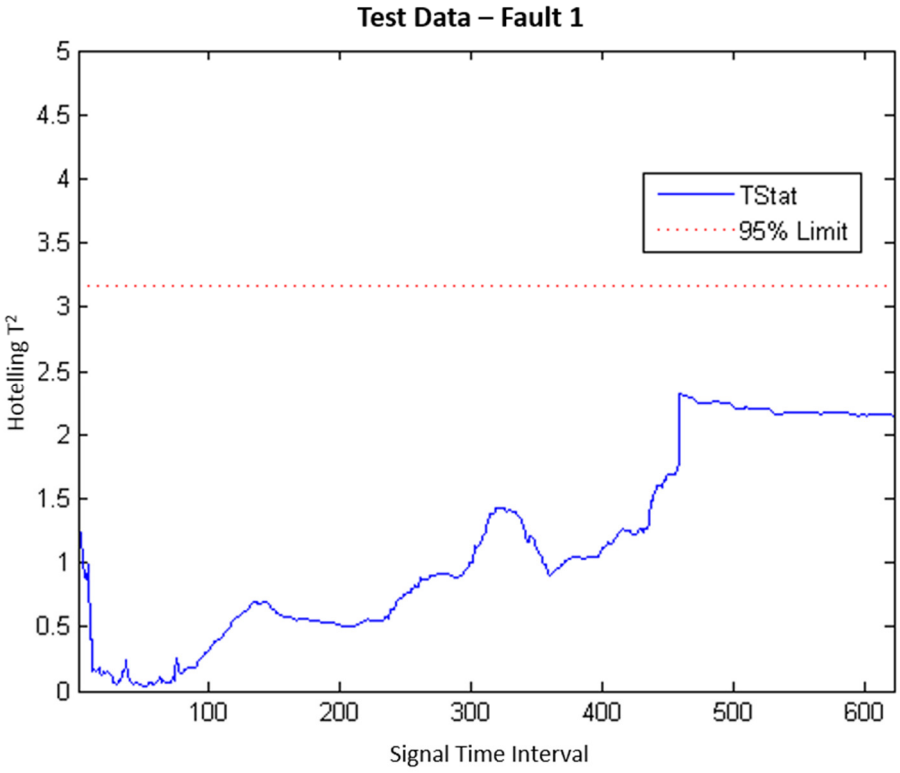


Figure B.6: NOC Test Data Monitoring Charts (Multiphase Models, γ = 20) – Case Study 1

a)



b)

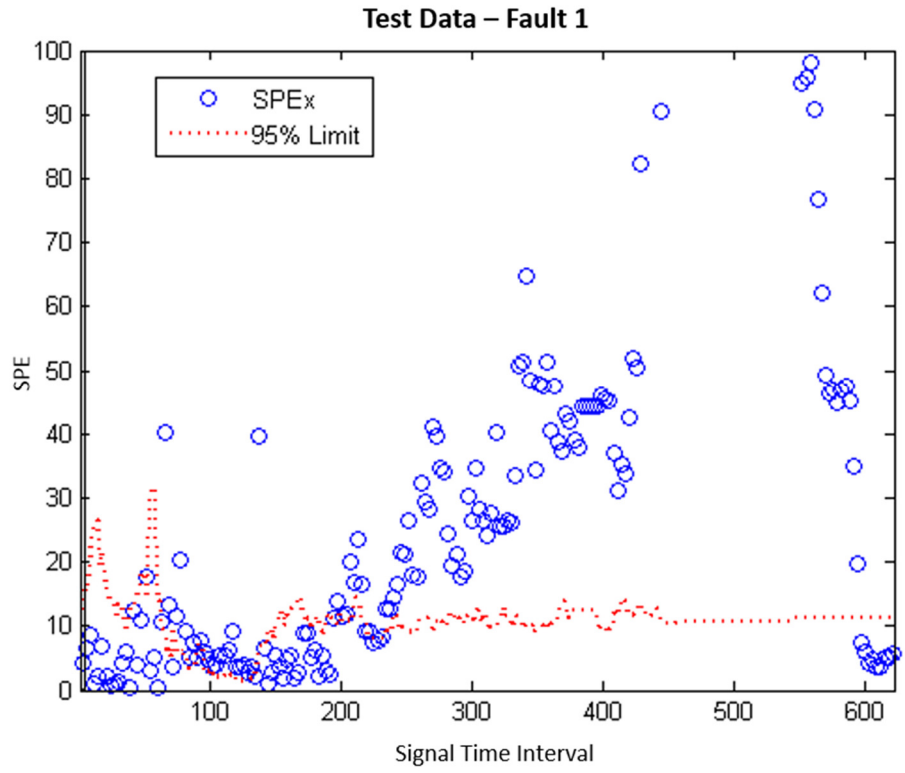
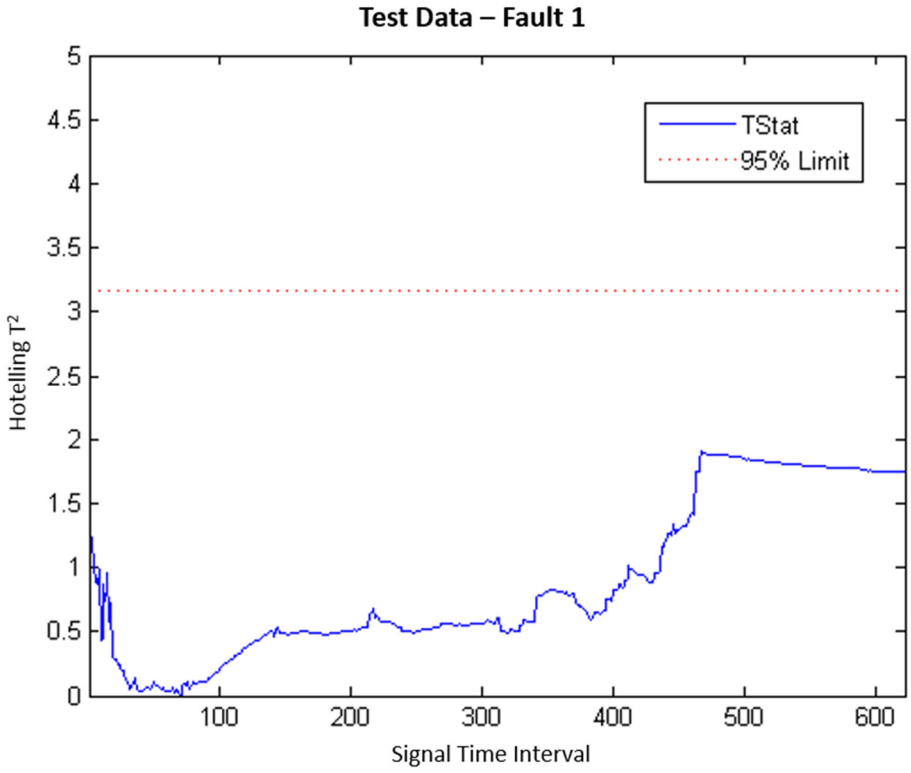


Figure B.7: Fault 1 Test Data Monitoring Charts (Global Model, γ = 5) – Case Study 1

a)



b)

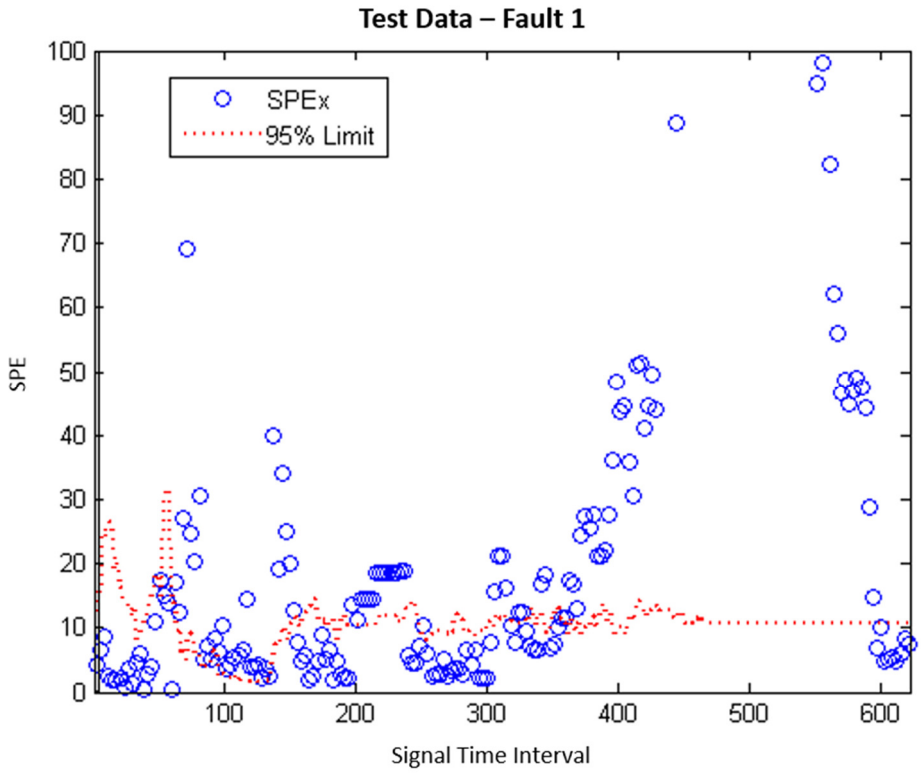
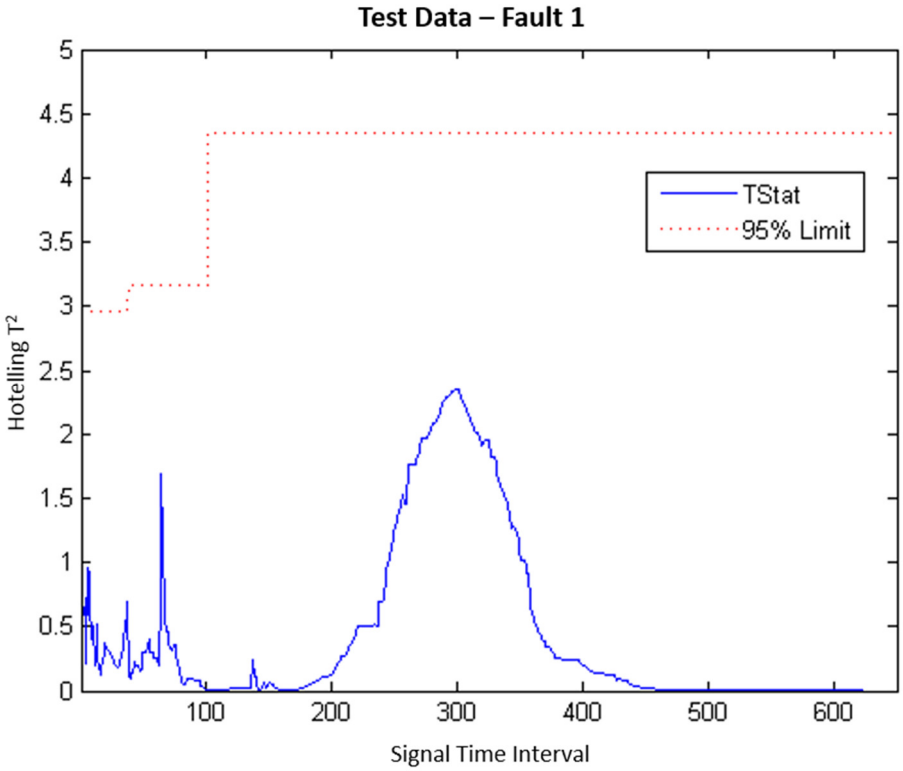


Figure B.8: Fault 1 Test Data Monitoring Charts (Global Model, γ = 20) – Case Study 1

a)



b)

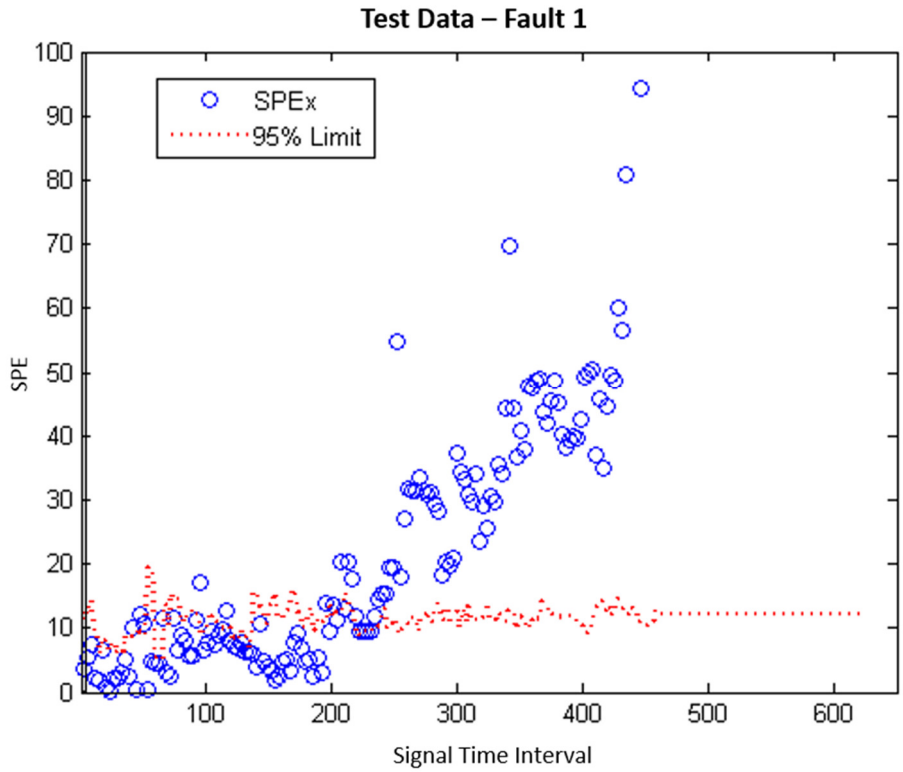
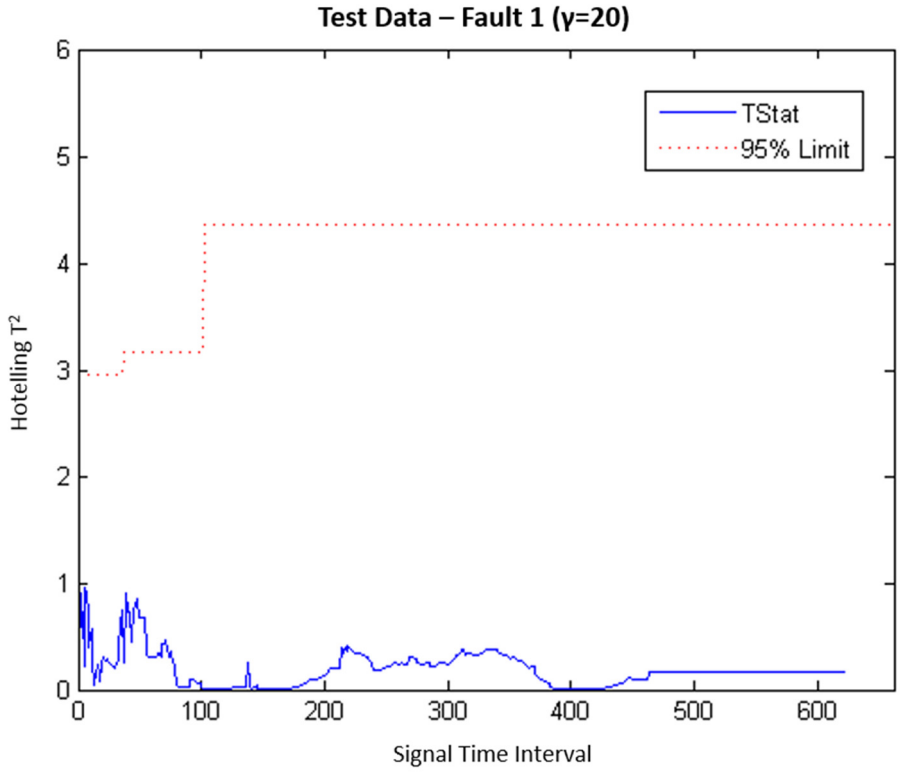


Figure B.9: Fault 1 Test Data Monitoring Charts (Operational Phase Models, γ = 5) – Case Study 1

a)



b)

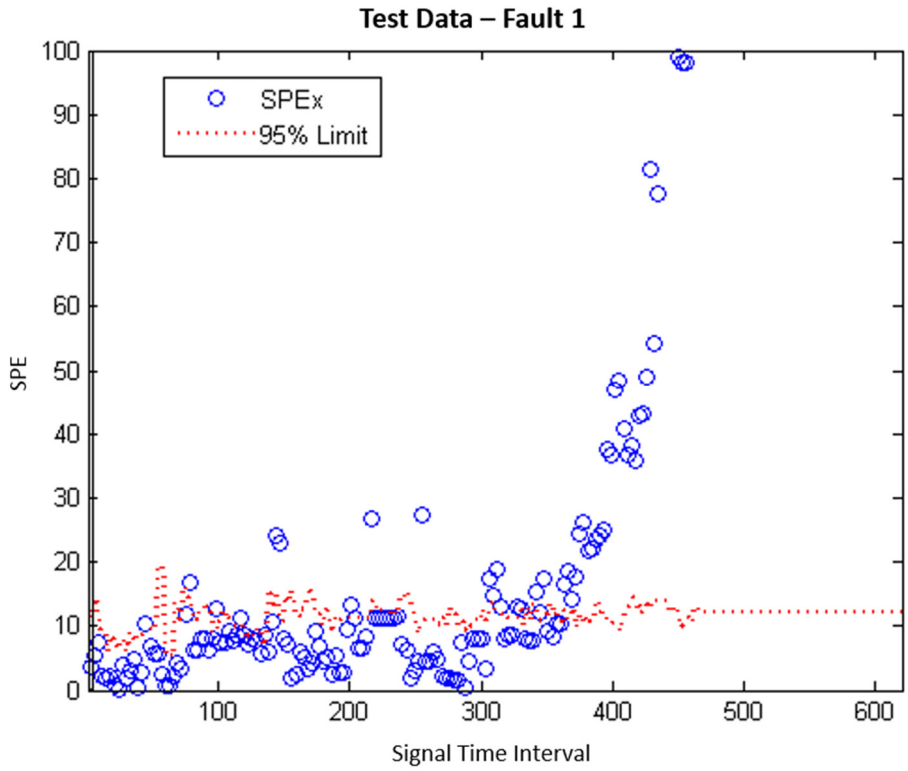
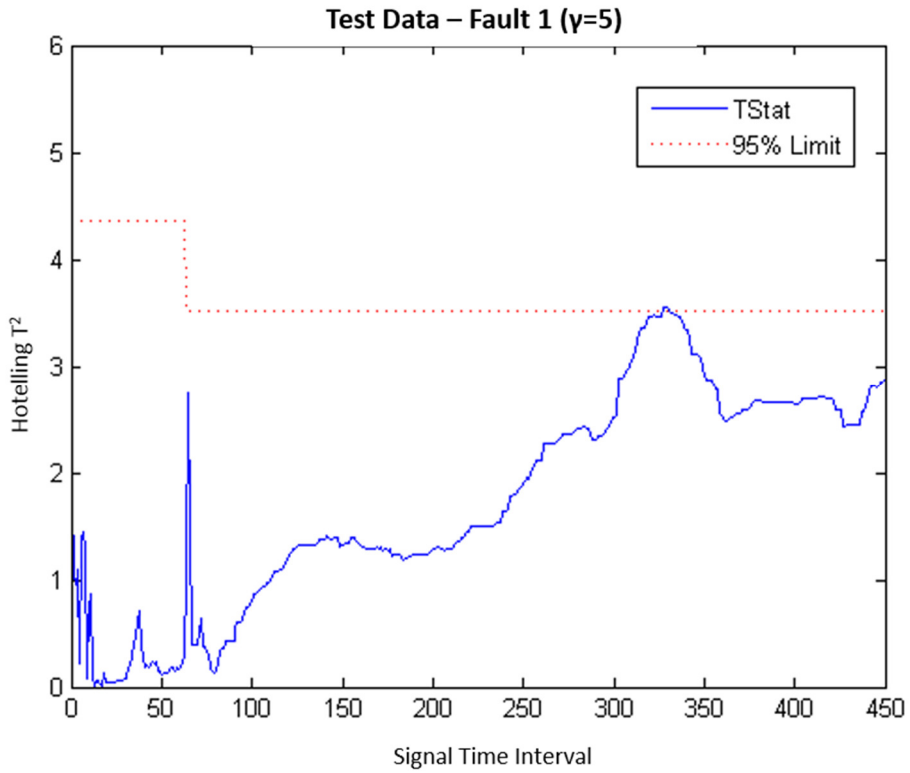


Figure B.10: Fault 1 Test Data Monitoring Charts (Operational Phase Models, γ = 20) – Case Study 1

a)



b)

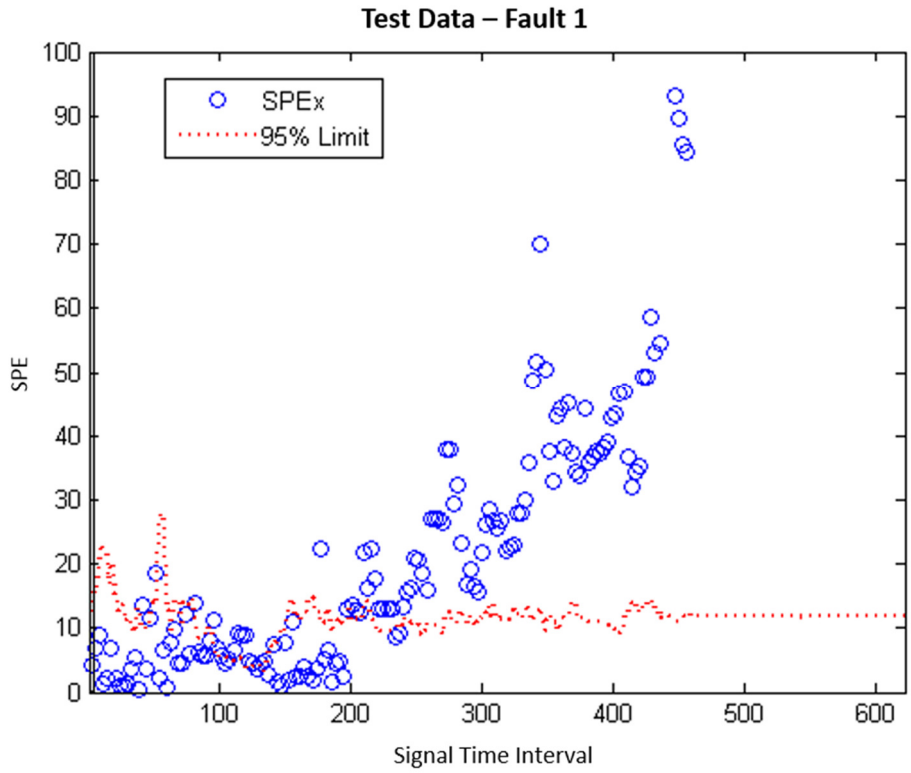
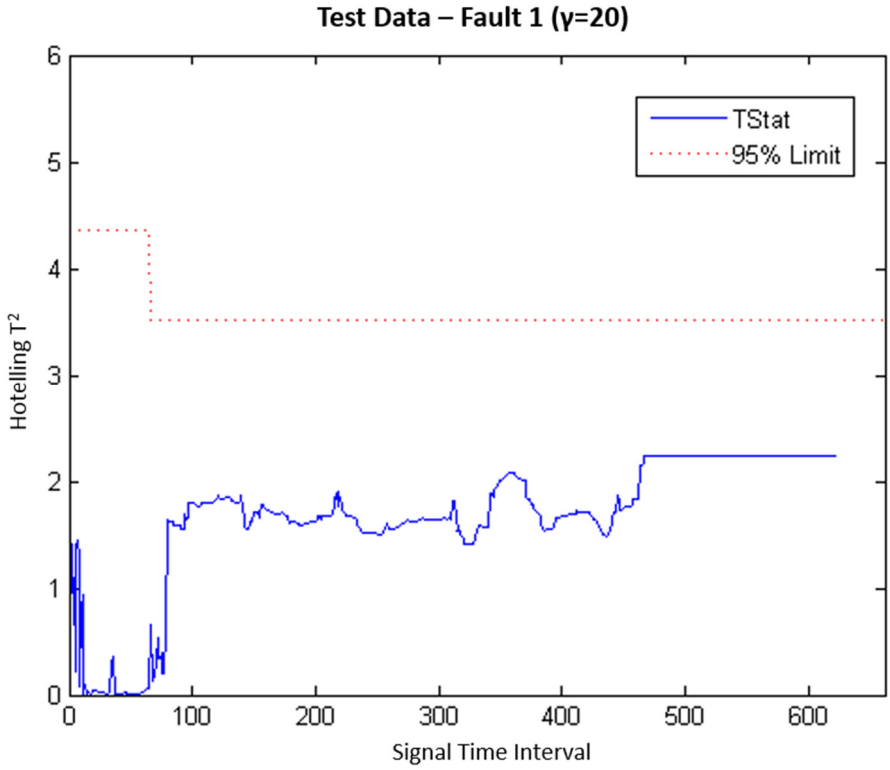


Figure B.11: Fault 1 Test Data Monitoring Charts (Multiphase Models, γ = 5) – Case Study 1

a)



b)

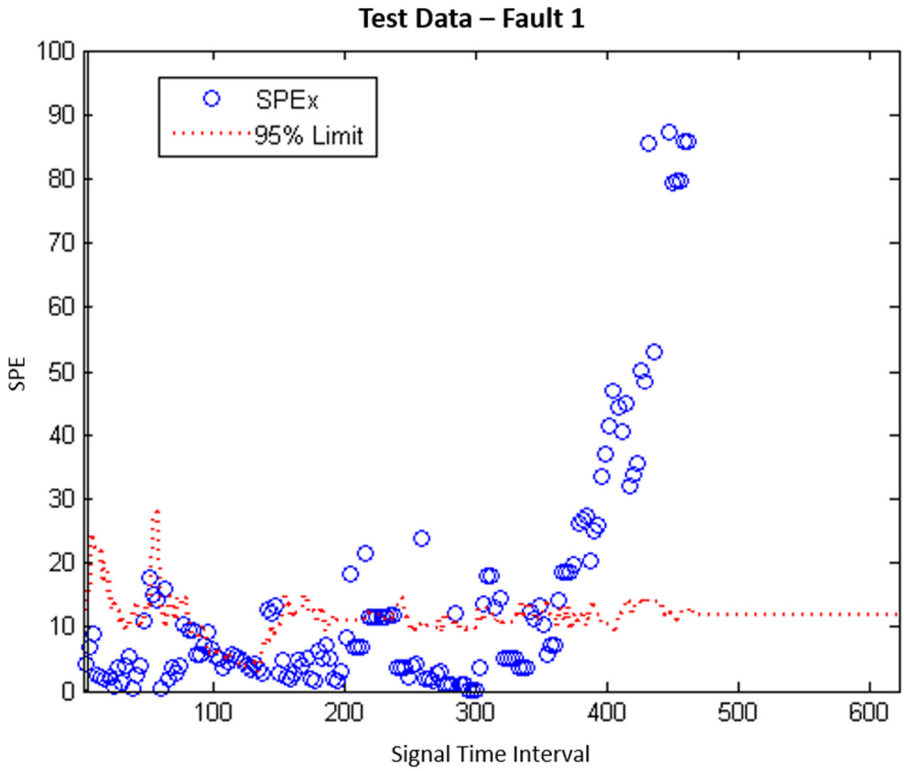
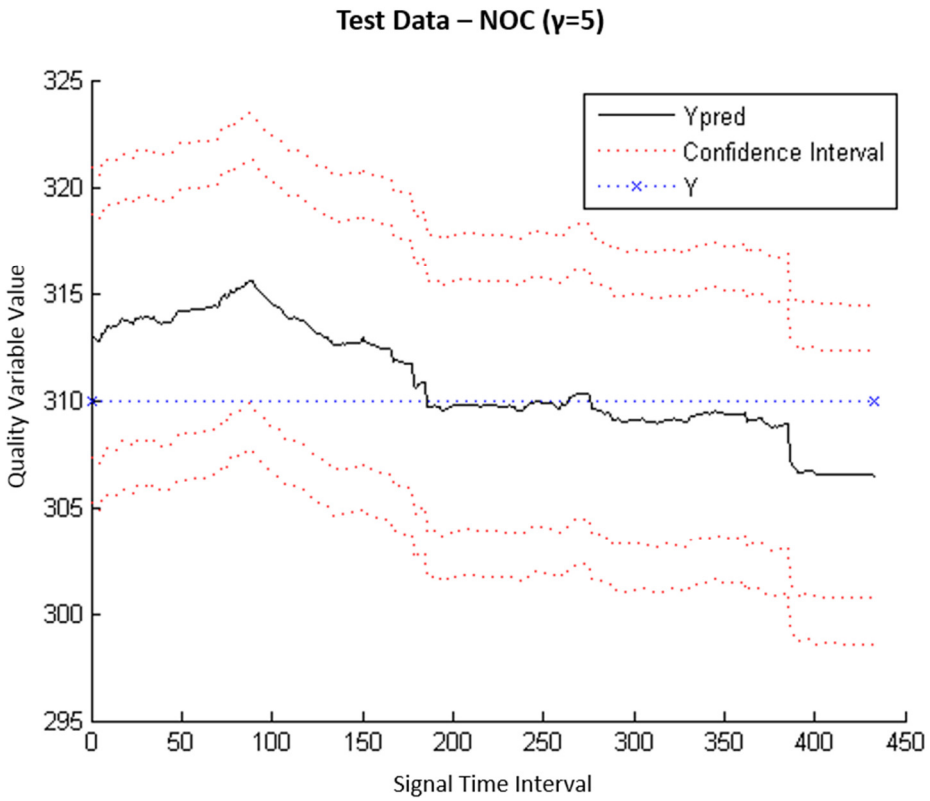


Figure B.12: Fault 1 Test Data Monitoring Charts (Multiphase Model, γ = 20) – Case Study 1

a)



b)

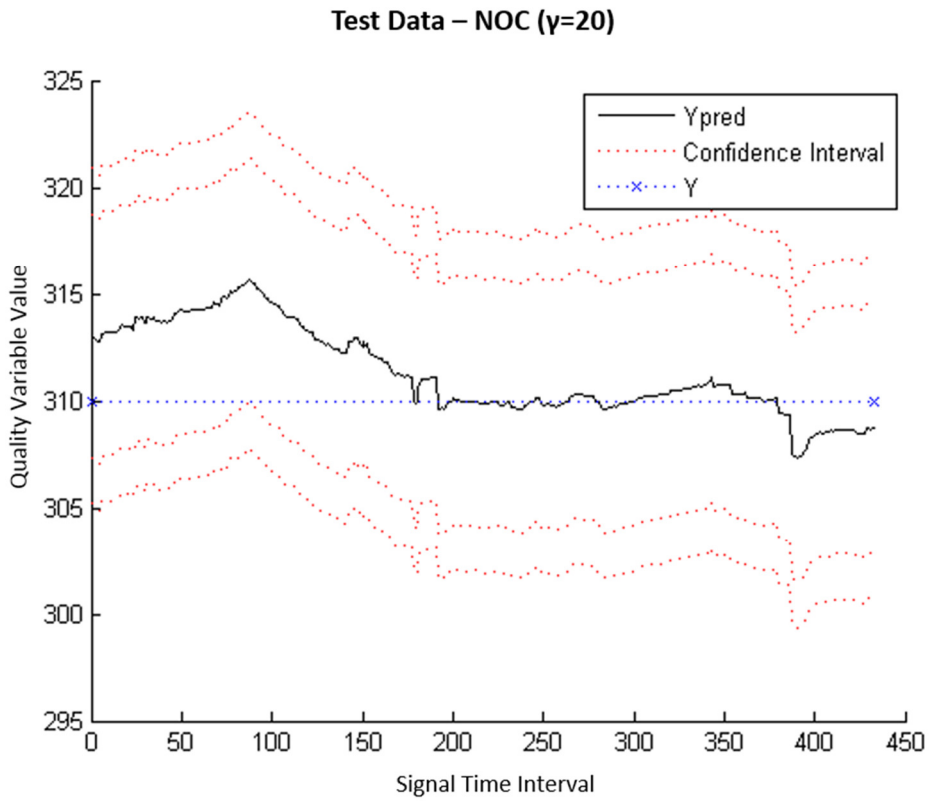
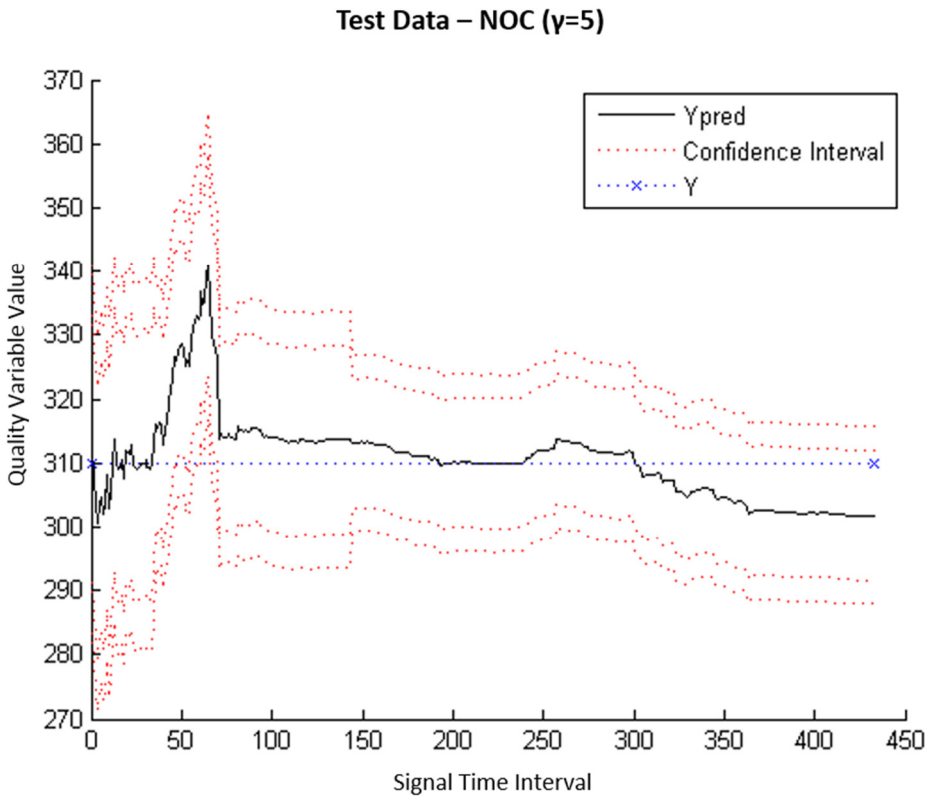


Figure B.13: NOC Test Data End-of-Batch Prediction (Global Model) – Case Study 1

a)



b)

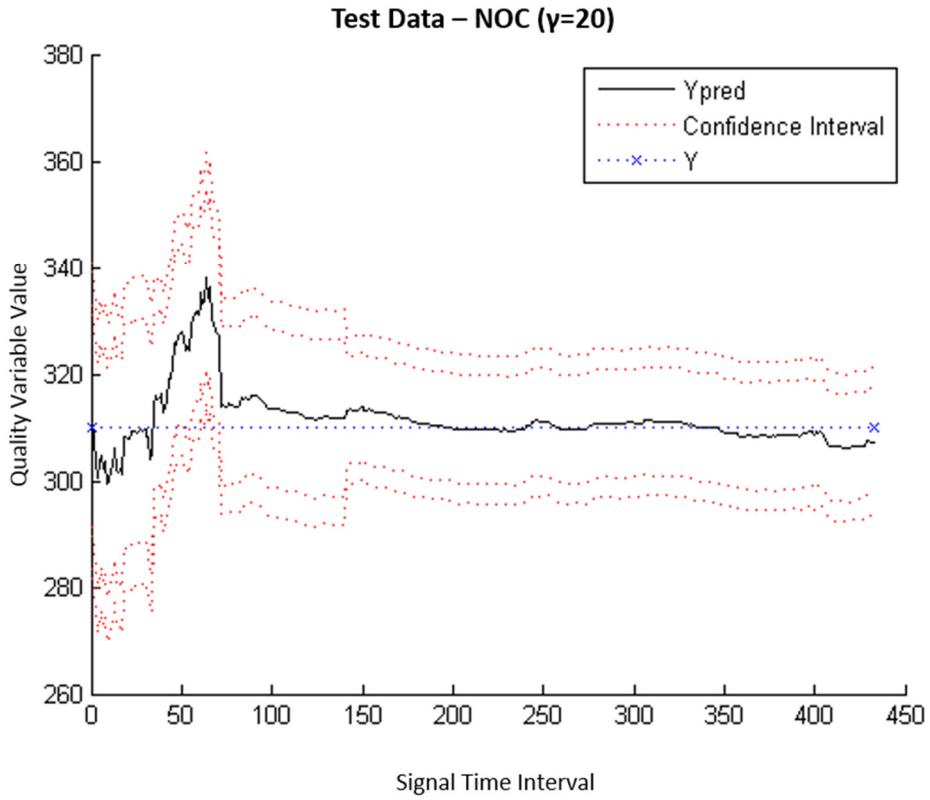
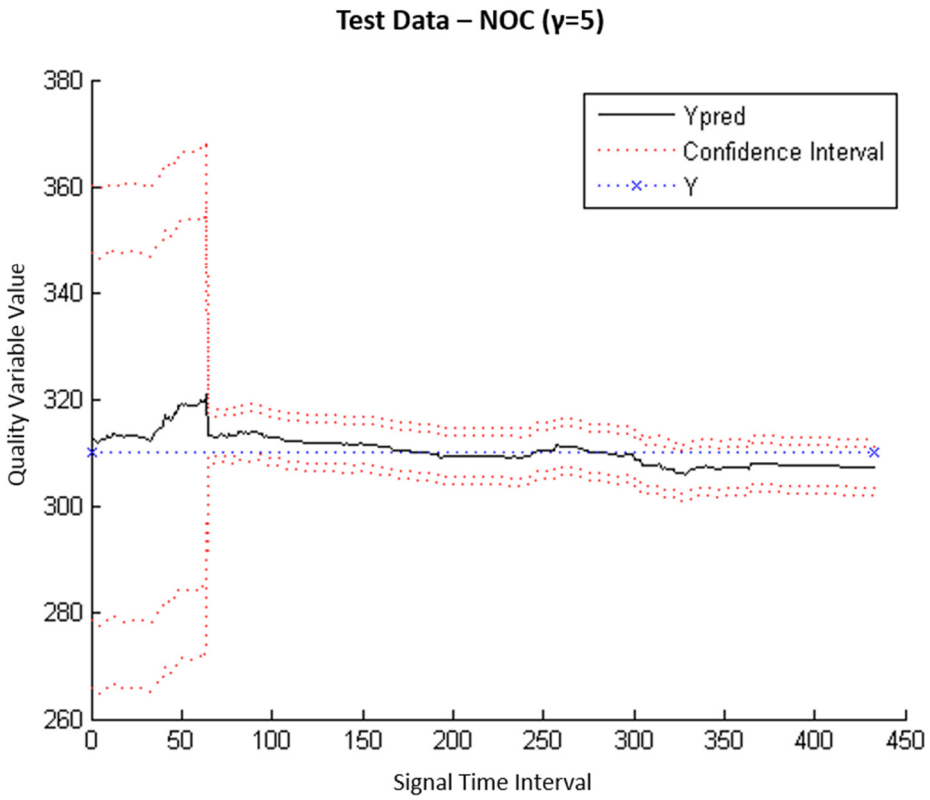


Figure B.14: NOC Test Data End-of-Batch Prediction (Operational Phase Model) – Case Study 1

a)



b)

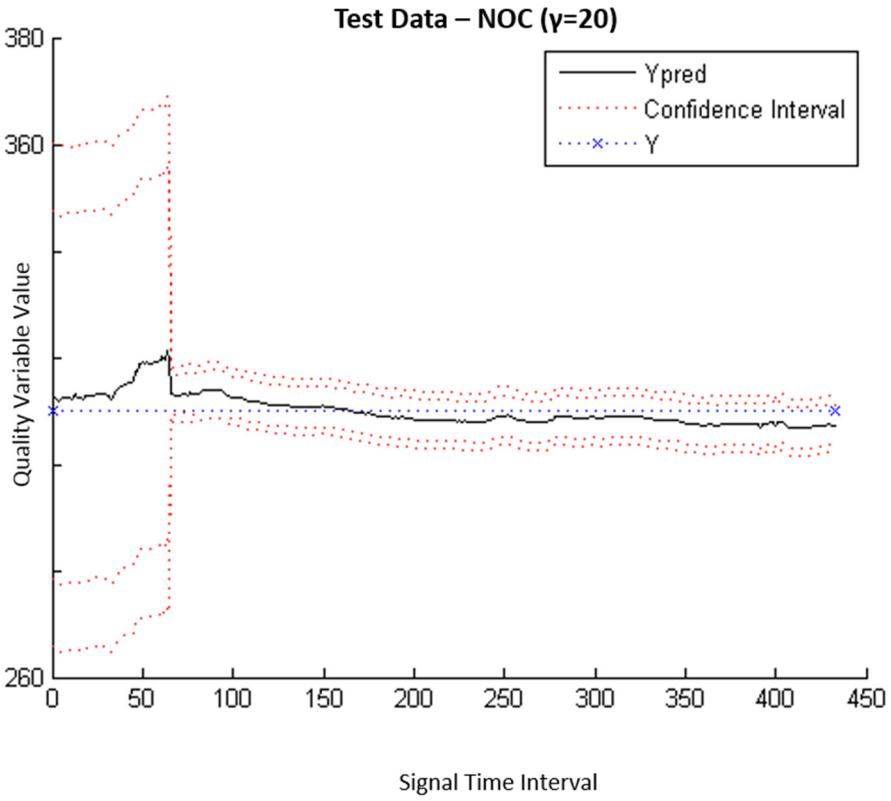
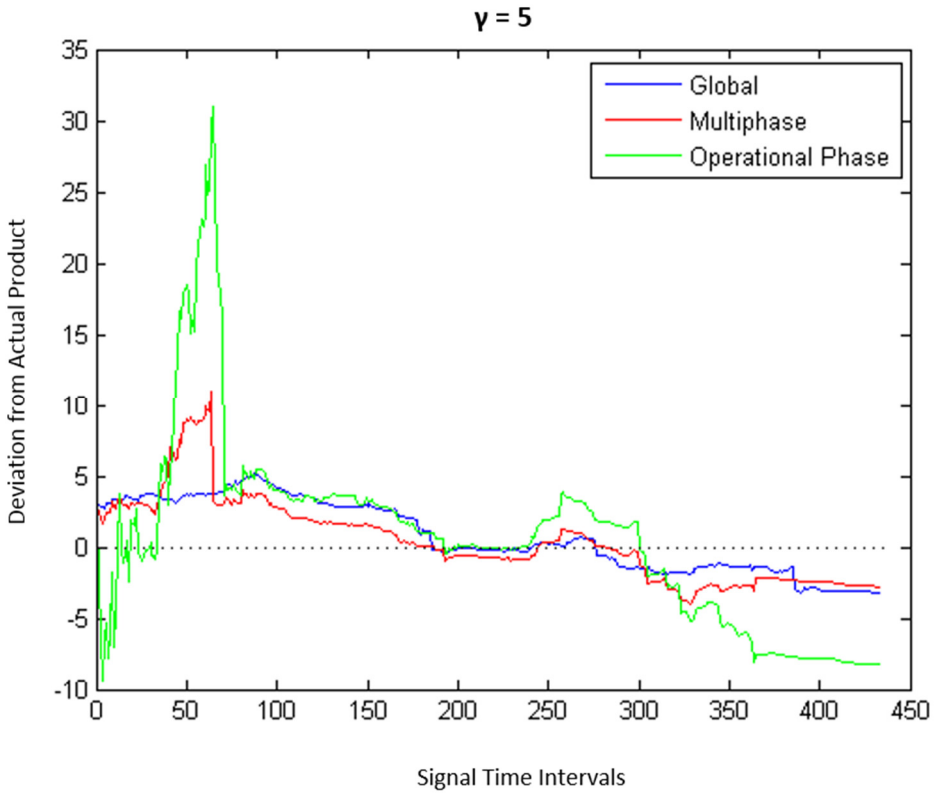


Figure B.15: NOC Test Data End-of-Batch Prediction (Multiphase Model) – Case Study 1

a)



b)

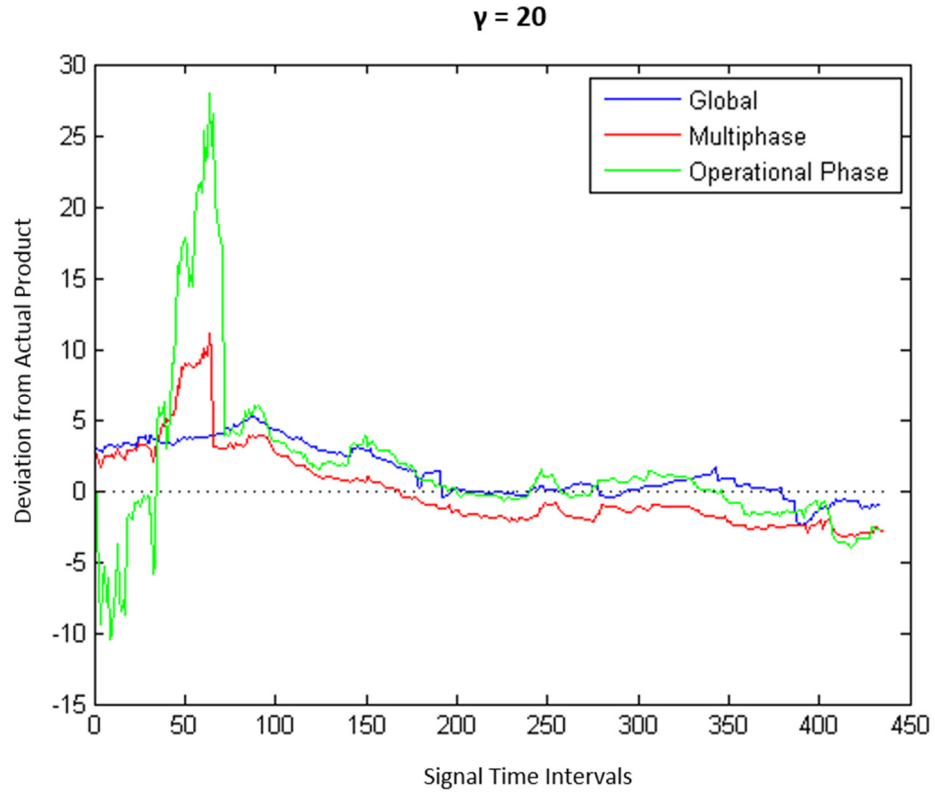
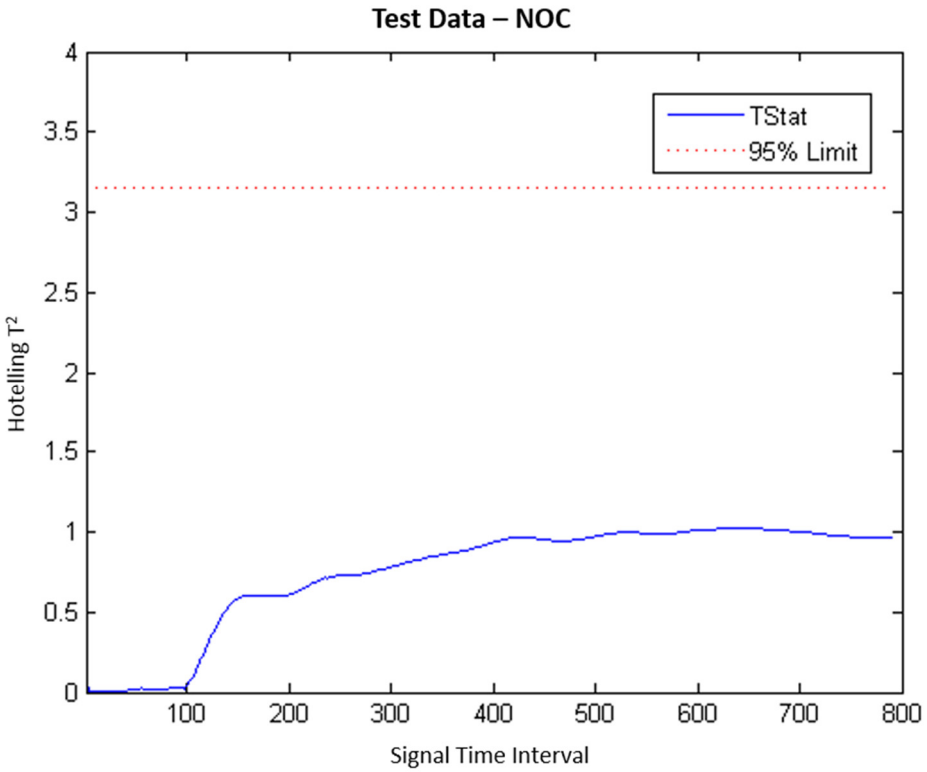


Figure B.16: Deviations from Actual Product Quality – Case Study 1

APPENDIX C: Penicillin Cultivation Process Monitoring Charts

a)



b)

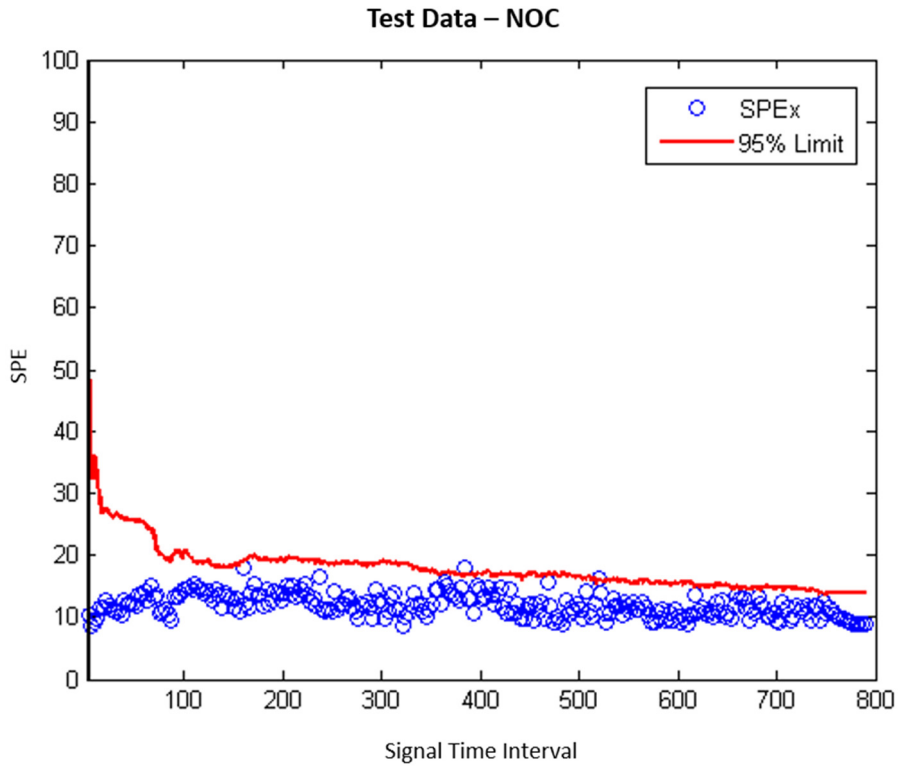
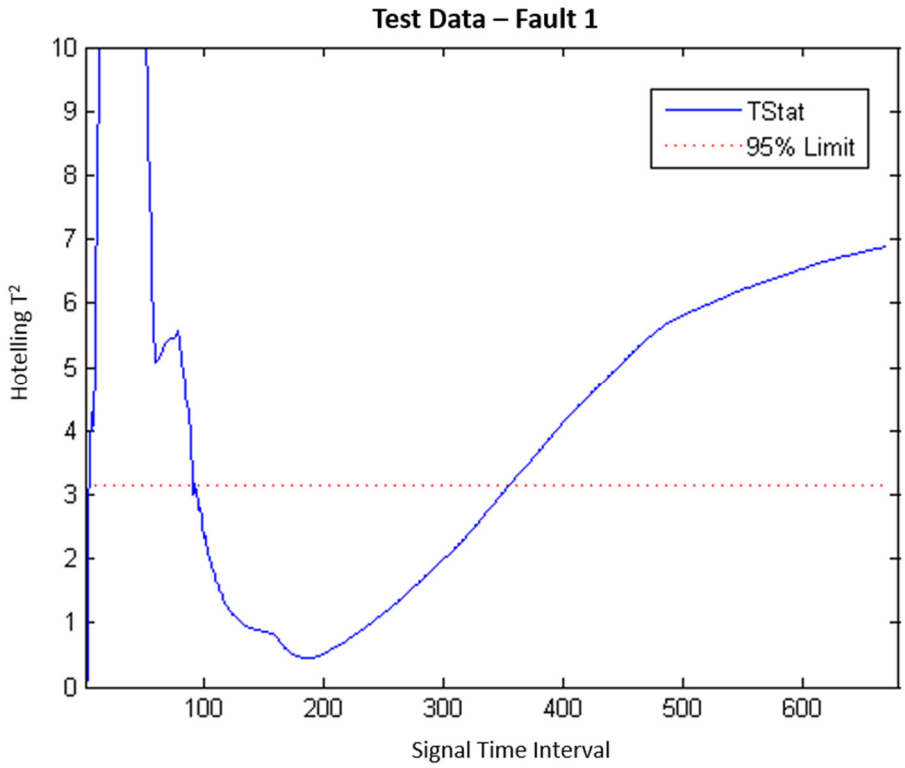


Figure C.1: NOC Test Data Monitoring Charts (Global Model, γ = 20) – Case Study 2

a)



b)

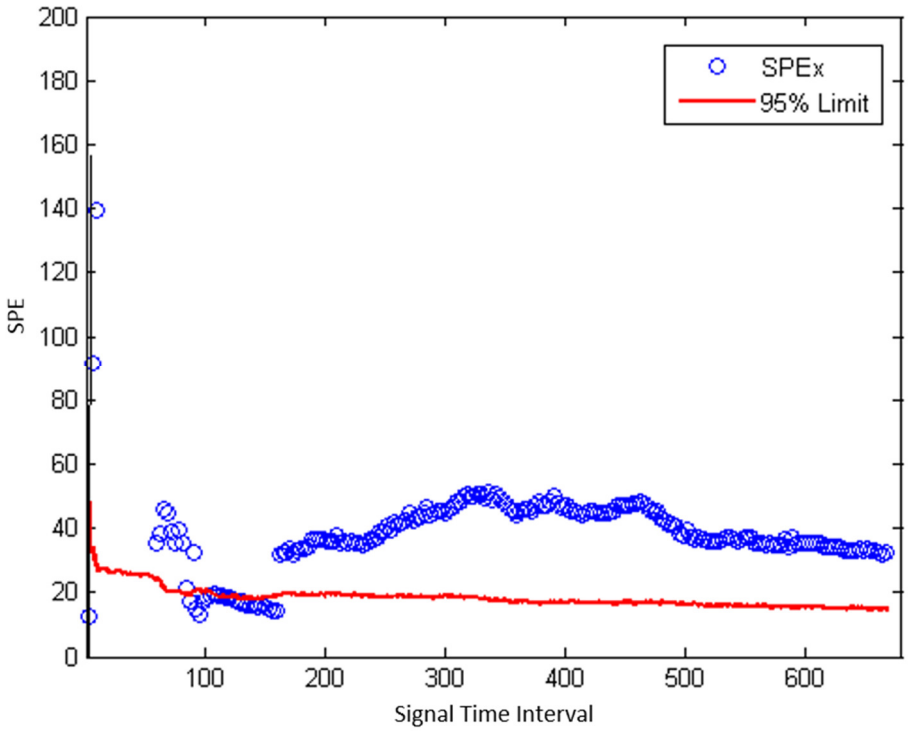
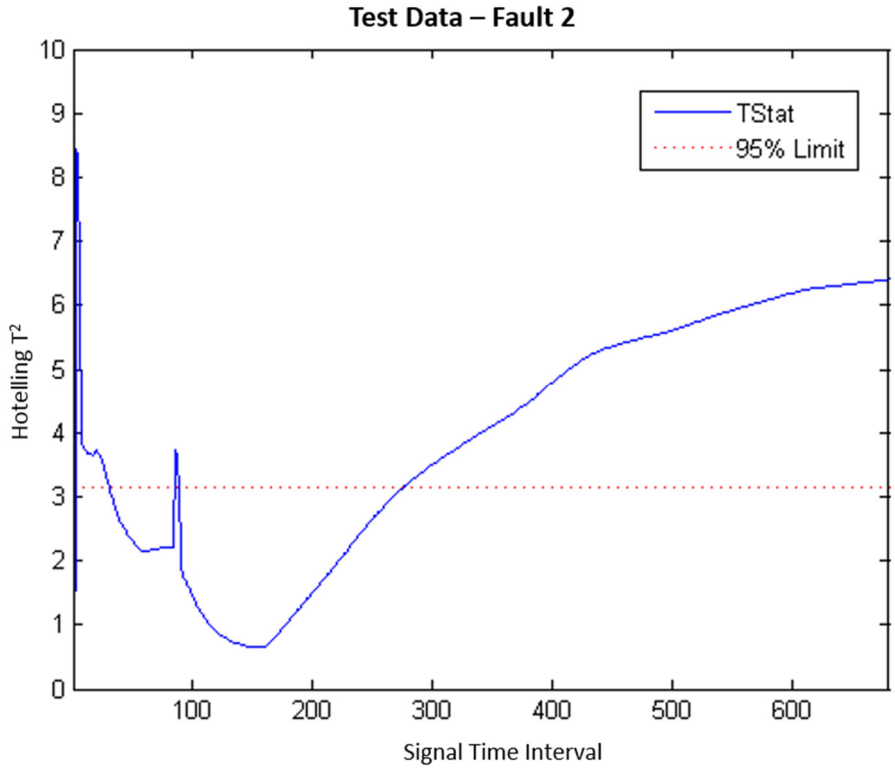


Figure C.2: Fault 1 Test Data Monitoring Charts (Global Model, γ = 20) – Case Study 2

a)



b)

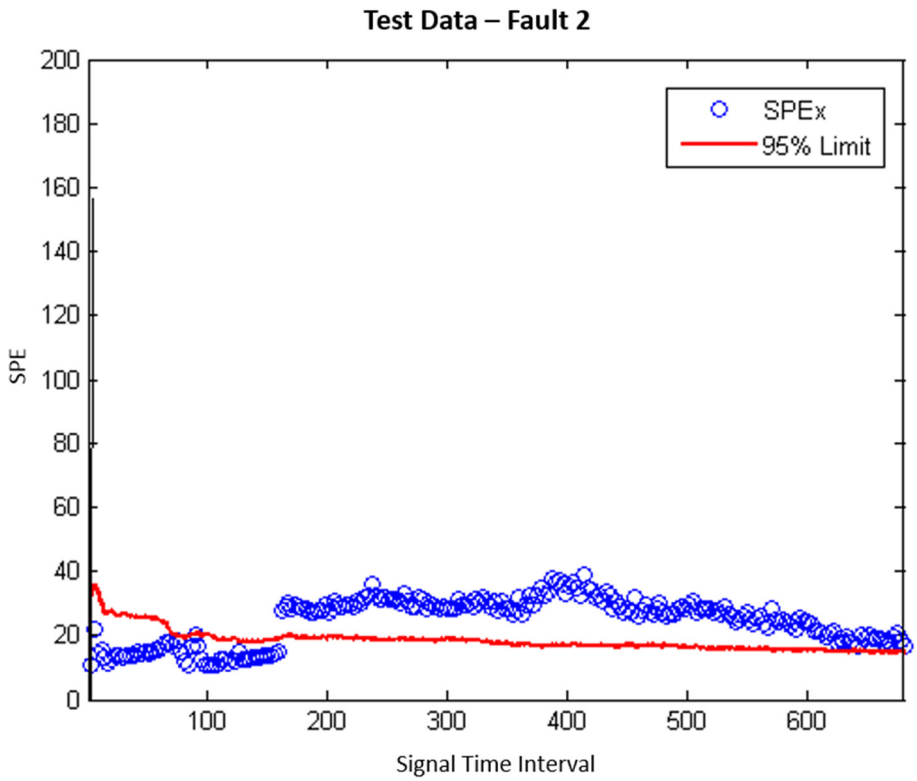
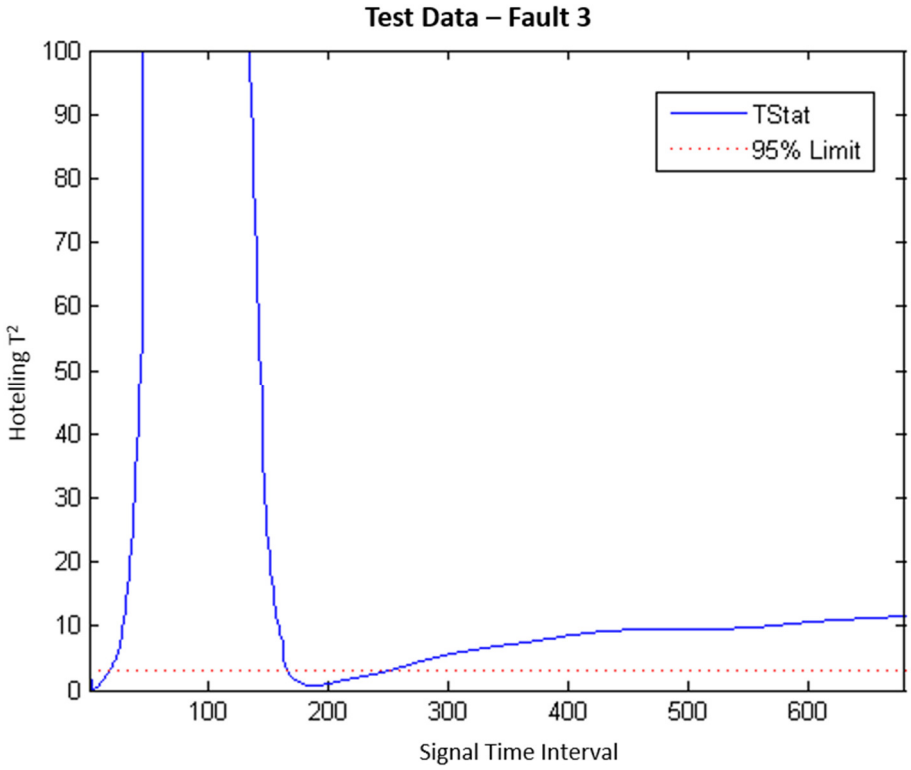


Figure C.3: Fault 2 Test Data Monitoring Charts (Global Model, γ = 20) – Case Study 2

a)



b)

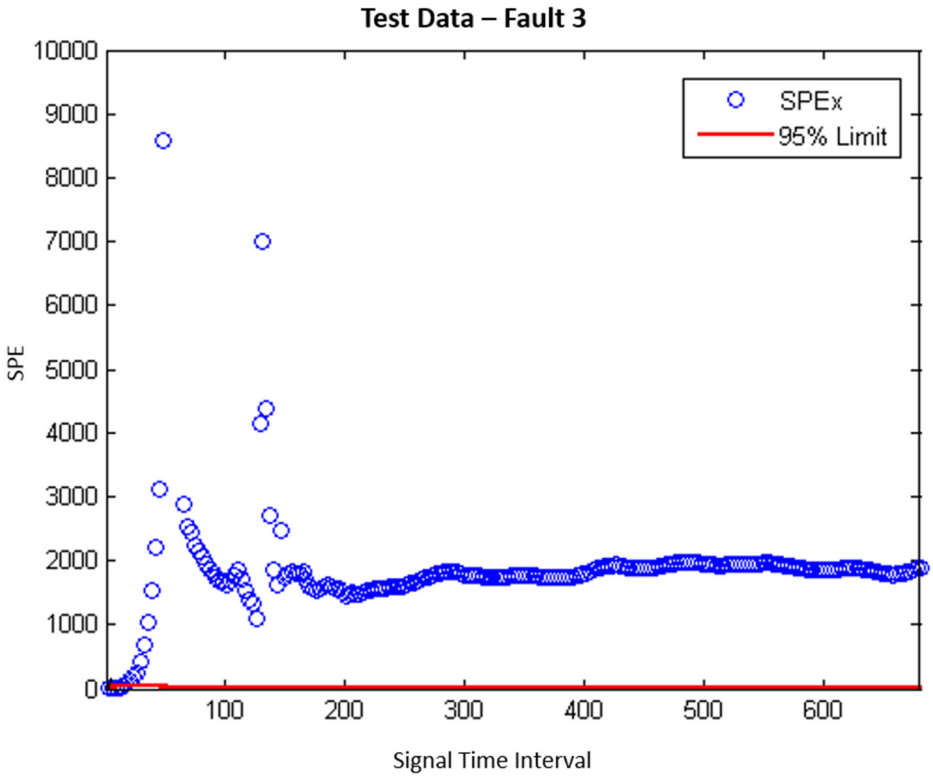
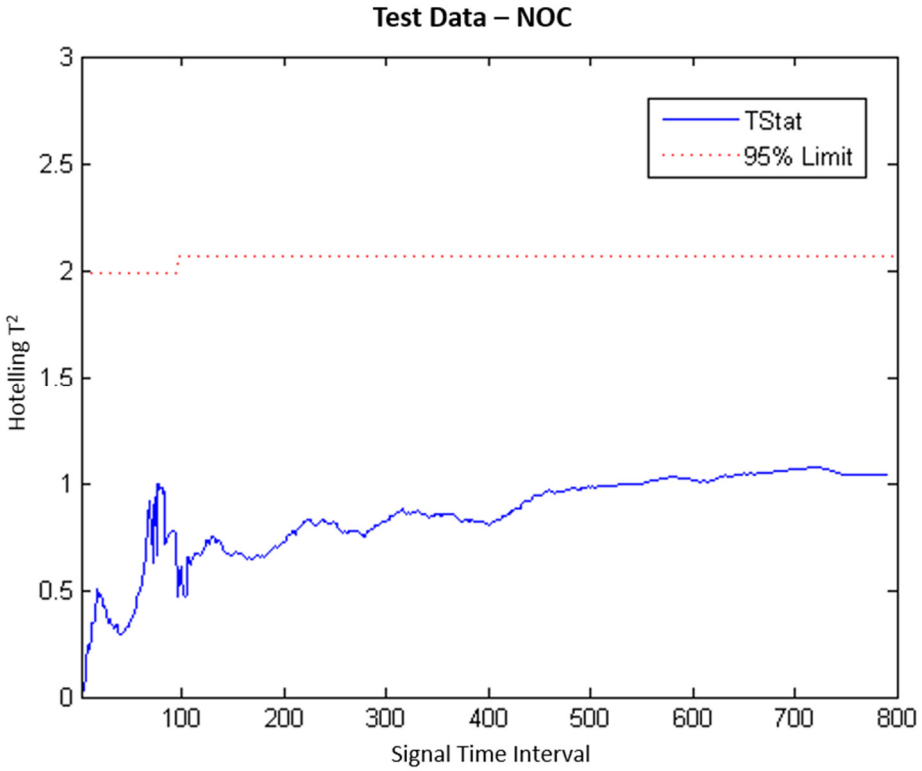


Figure C.4: Fault 3 Test Data Monitoring Charts (Global Model, γ = 20) – Case Study 2

a)



b)

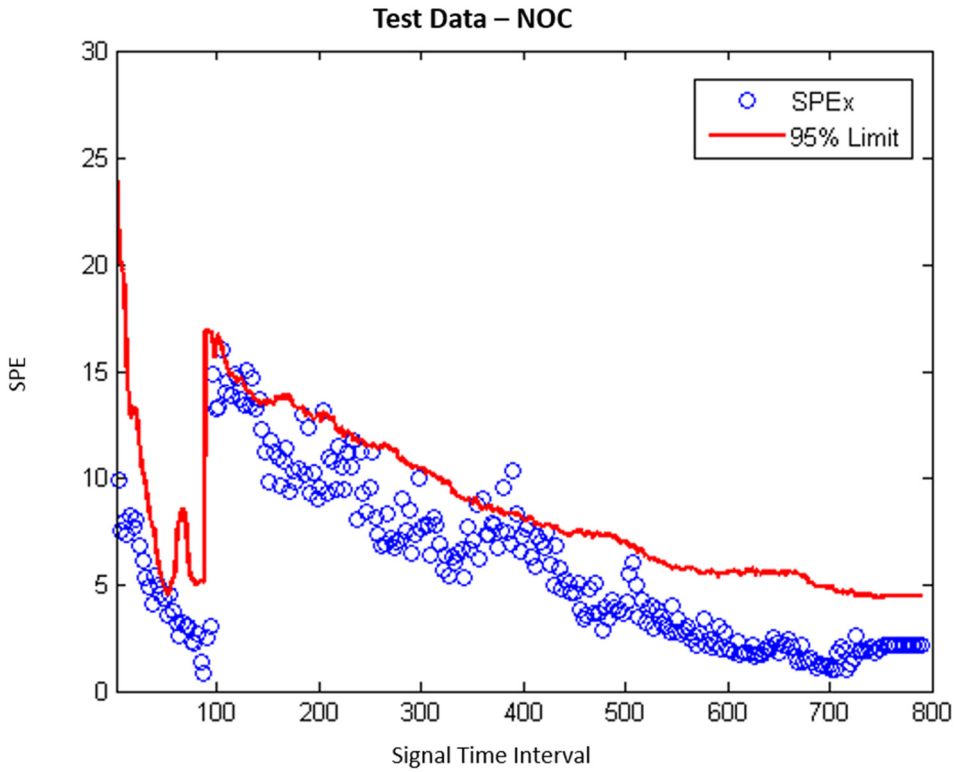
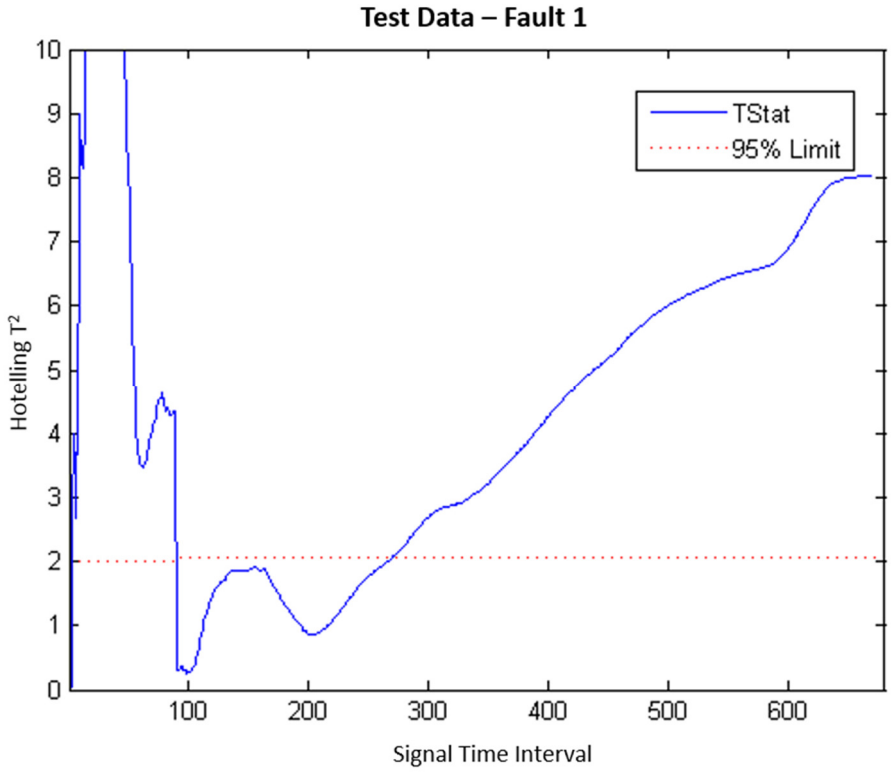


Figure C.5: NOC Test Data Monitoring Charts (Operational Model, γ = 20) – Case Study 2

a)



b)

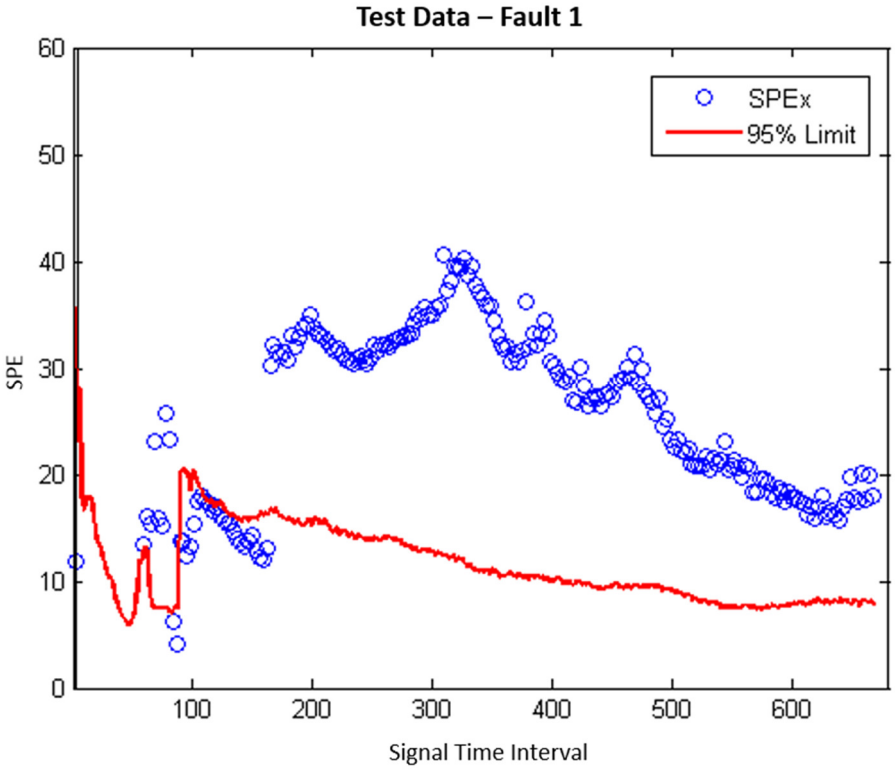
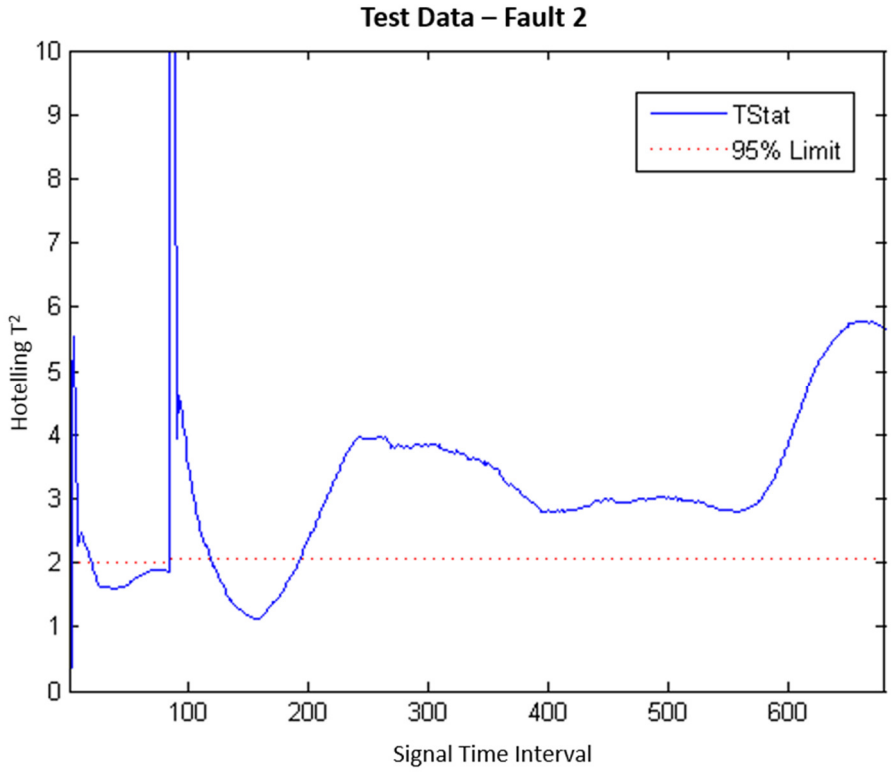


Figure C.6: Fault 1 Test Data Monitoring Charts (Operational Phase Model, γ = 20) – Case Study 2

a)



b)

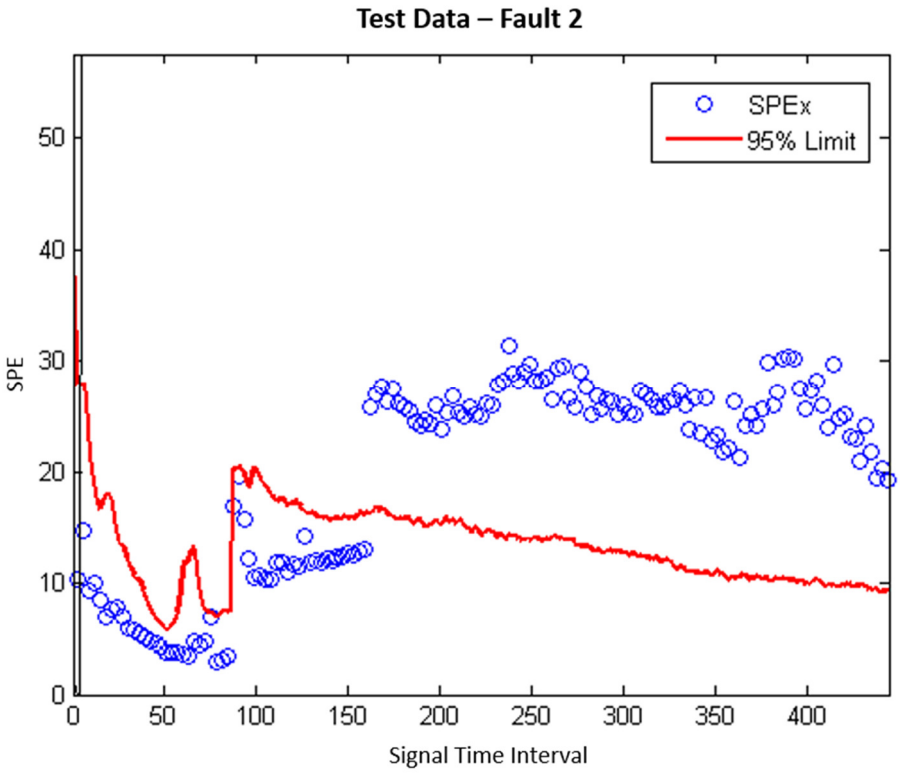
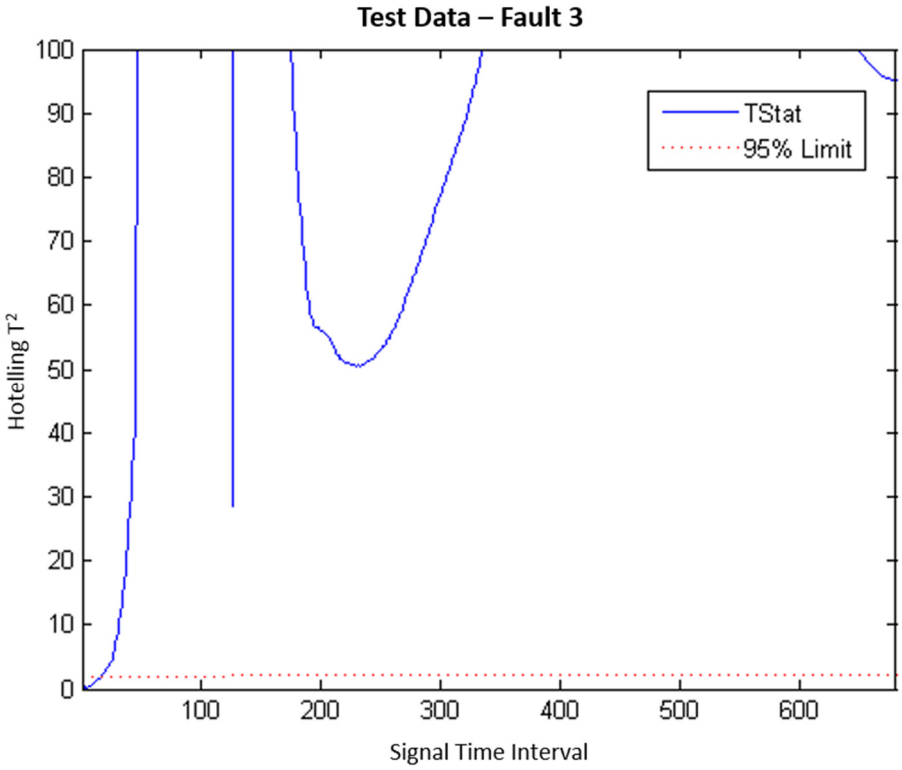


Figure C.7: Fault 2 Test Data Monitoring Charts (Operational Phase Models, γ = 20) – Case Study 2

a)



b)

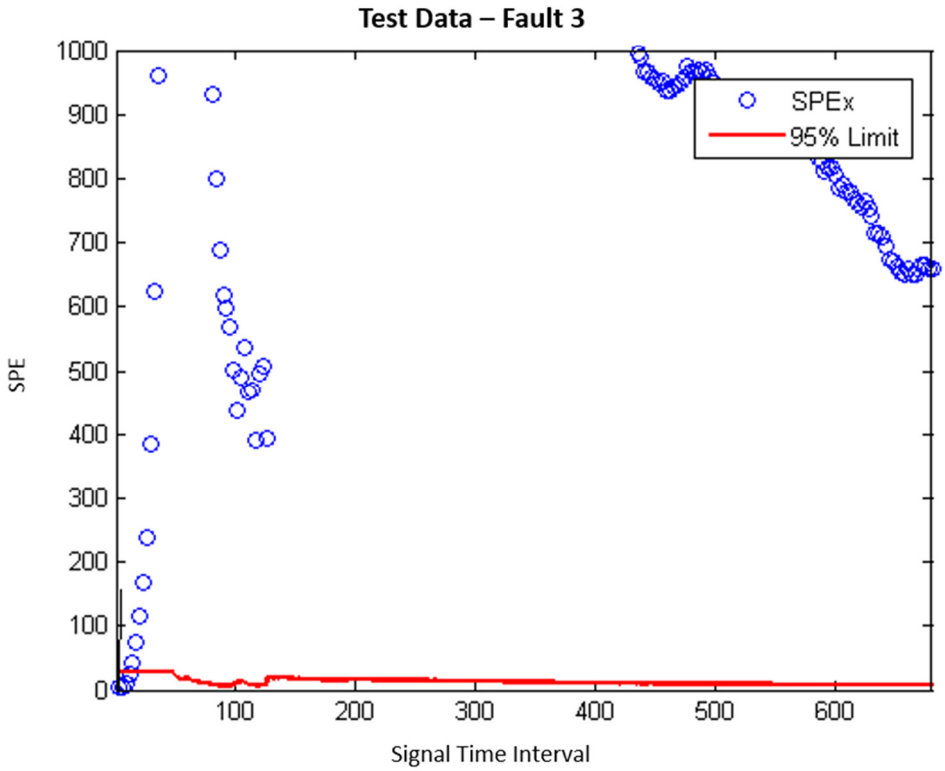
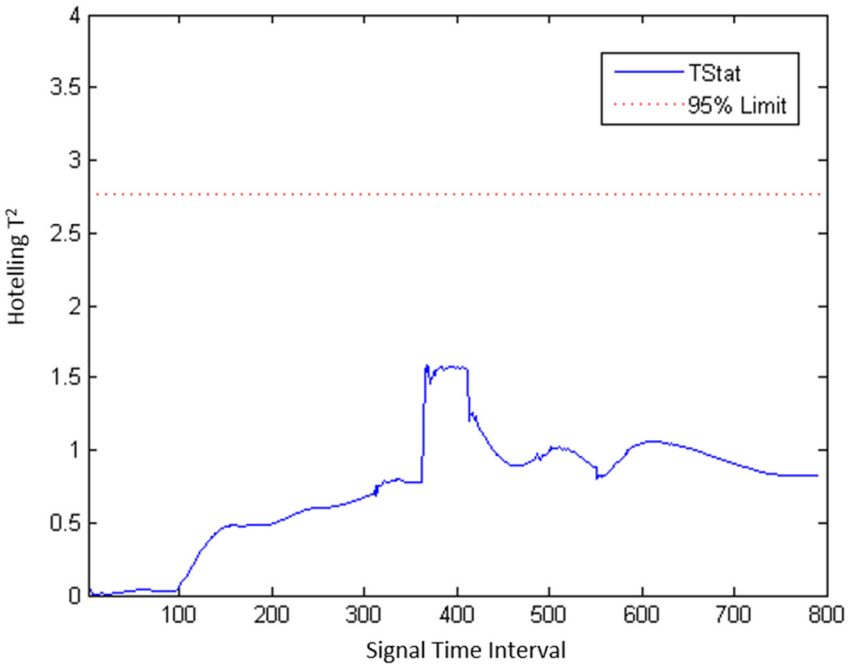


Figure C.8: Fault 3 Test Data Monitoring Charts (Operational Model, γ = 20) – Case Study 2

a)



b)

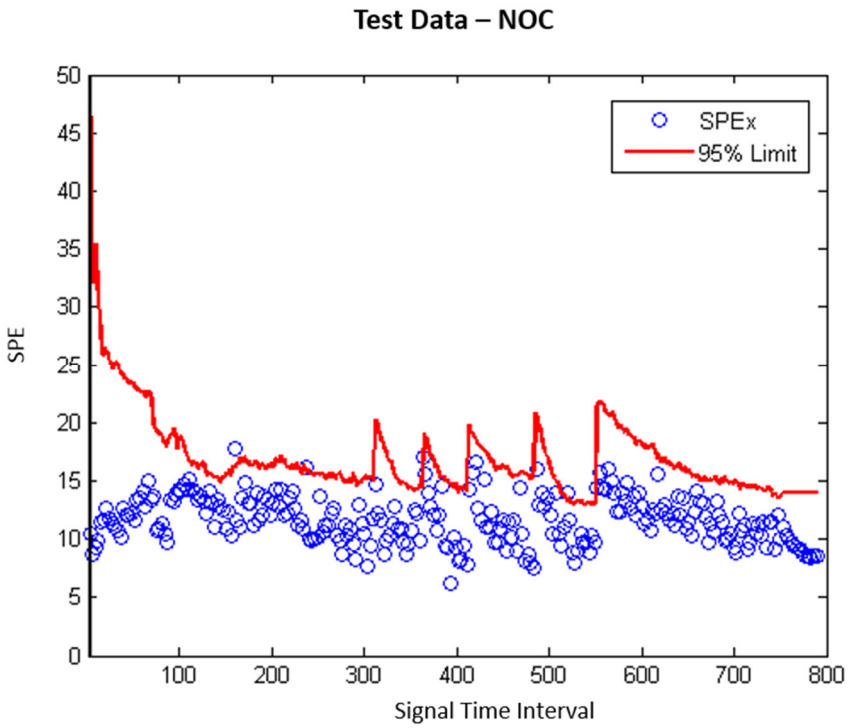
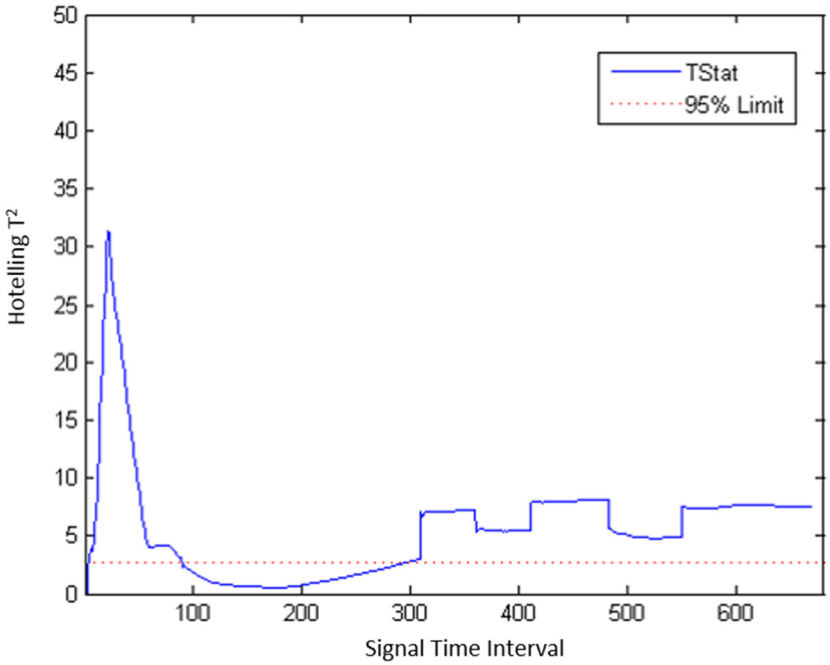


Figure C.9: NOC Test Data Monitoring Charts (Multiphase Model, γ = 20) – Case Study 2

a)



b)

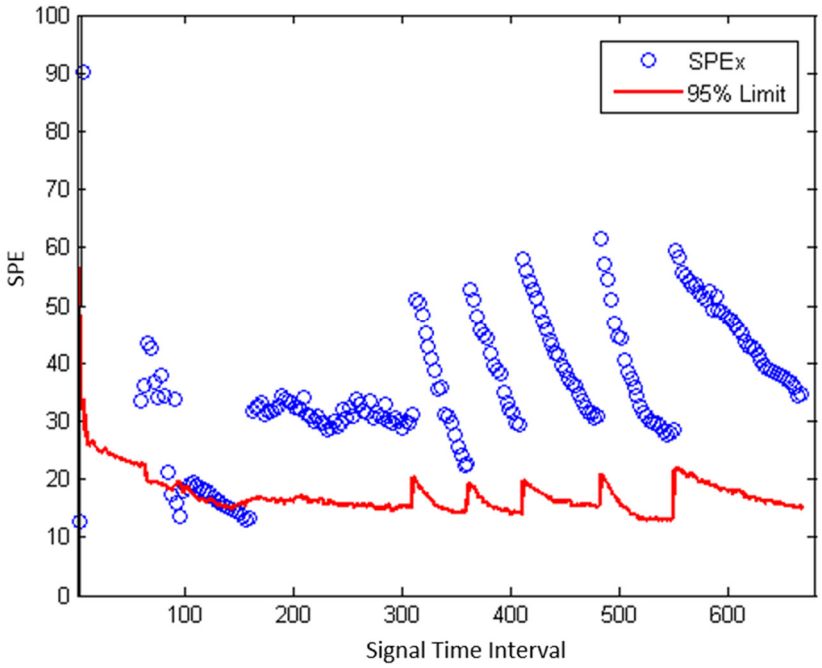
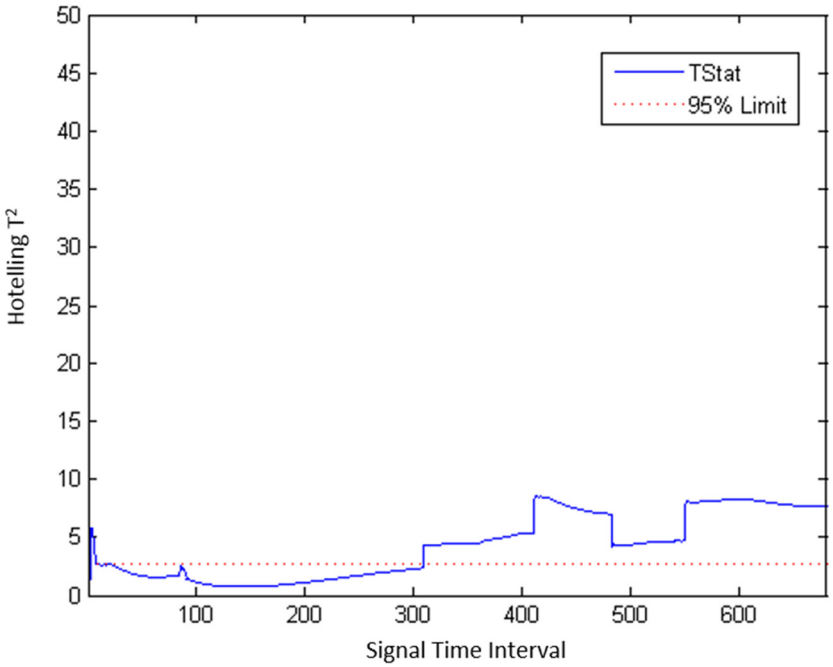


Figure C.10: Fault 1 Test Data Monitoring Charts (Multiphase Model, γ = 20) – Case Study 2

a)



b)

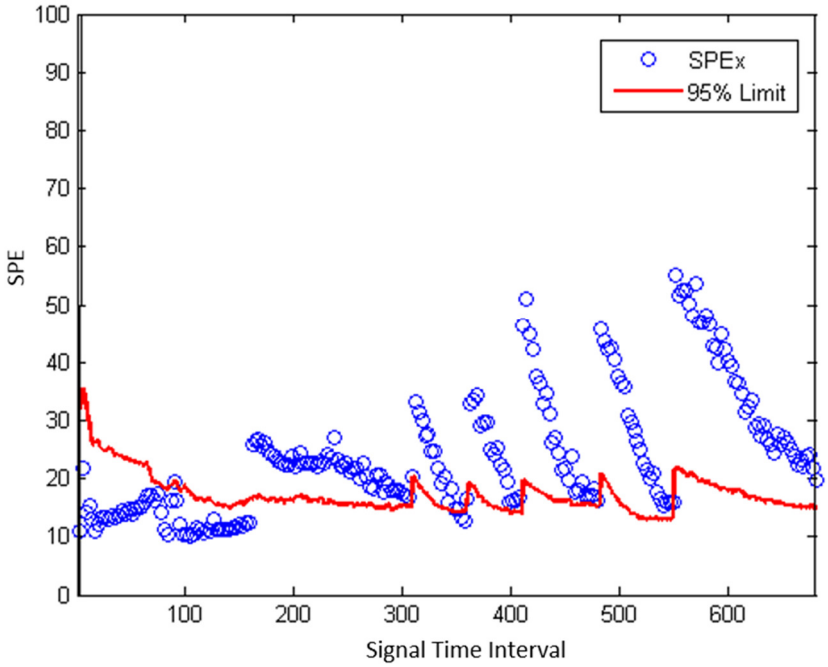
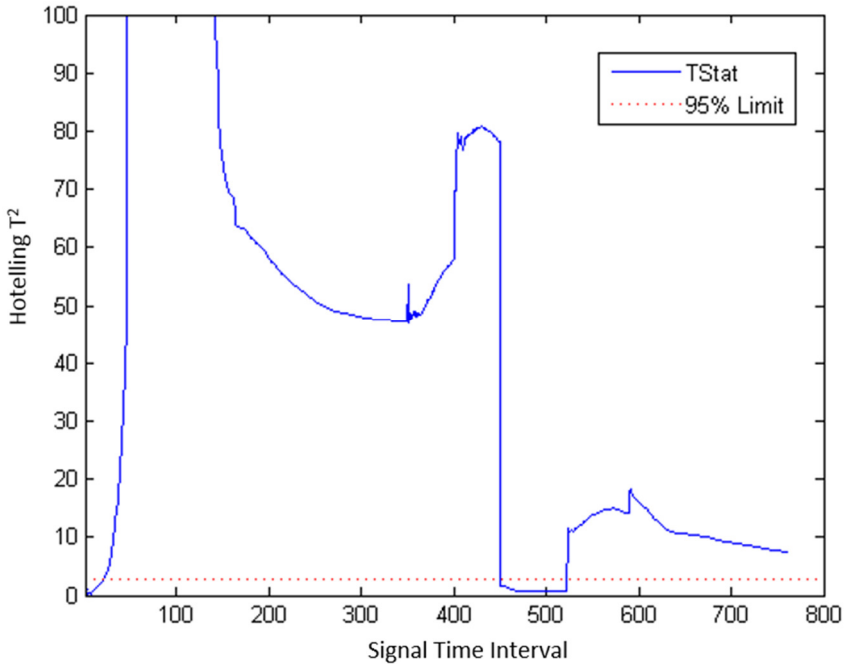


Figure C.11: Fault 2 Test Data Monitoring Charts (Multiphase Phase Models, γ = 20) – Case Study 2

a)



b)

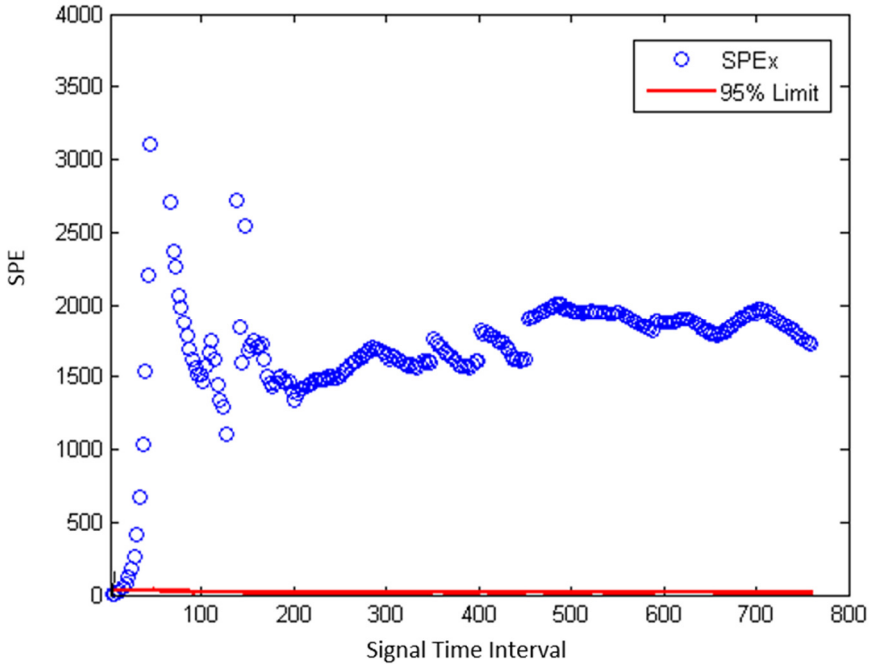


Figure C.12: Fault 3 Test Data Monitoring Charts (Multiphase Model, γ = 20) – Case Study 2

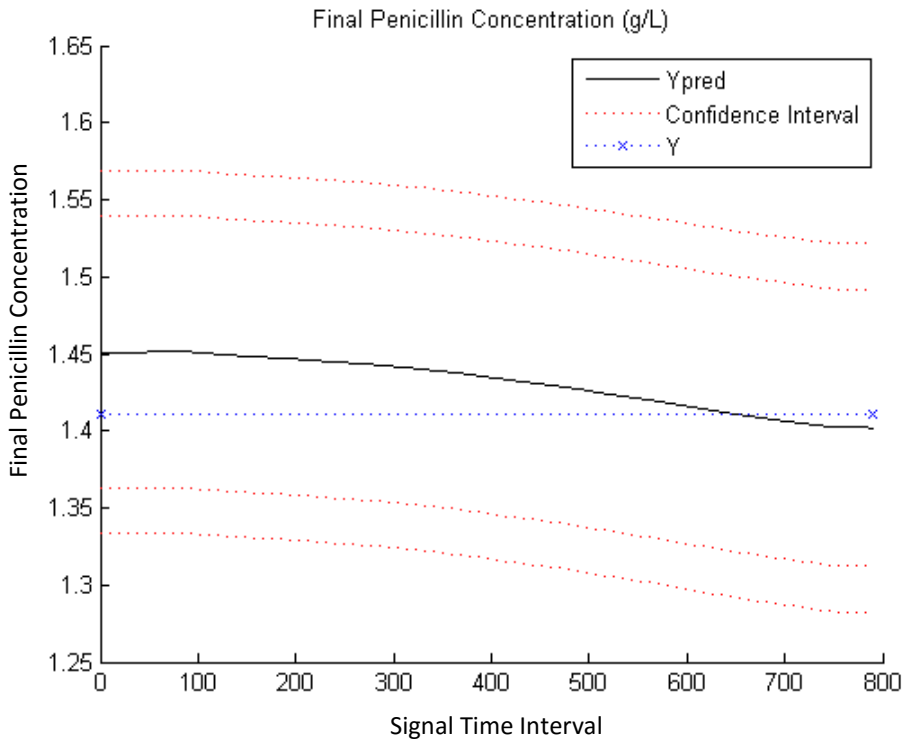


Figure C.13: Final Penicillin Concentration (Global Model) – Case Study 2

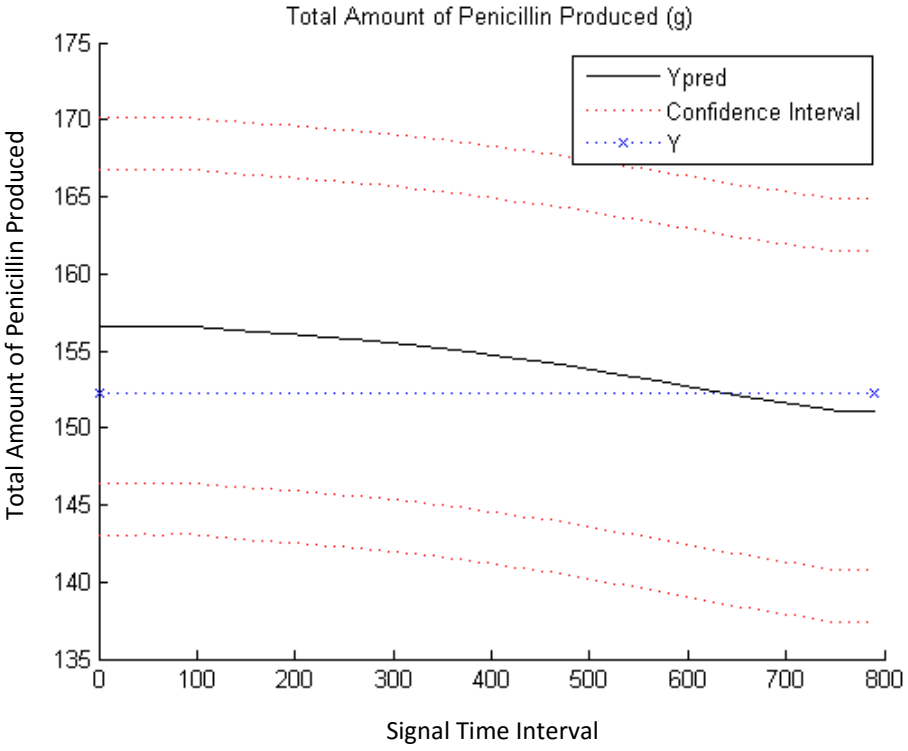


Figure C.14: Total Amount of Penicillin Produced (Global Model) – Case Study 2

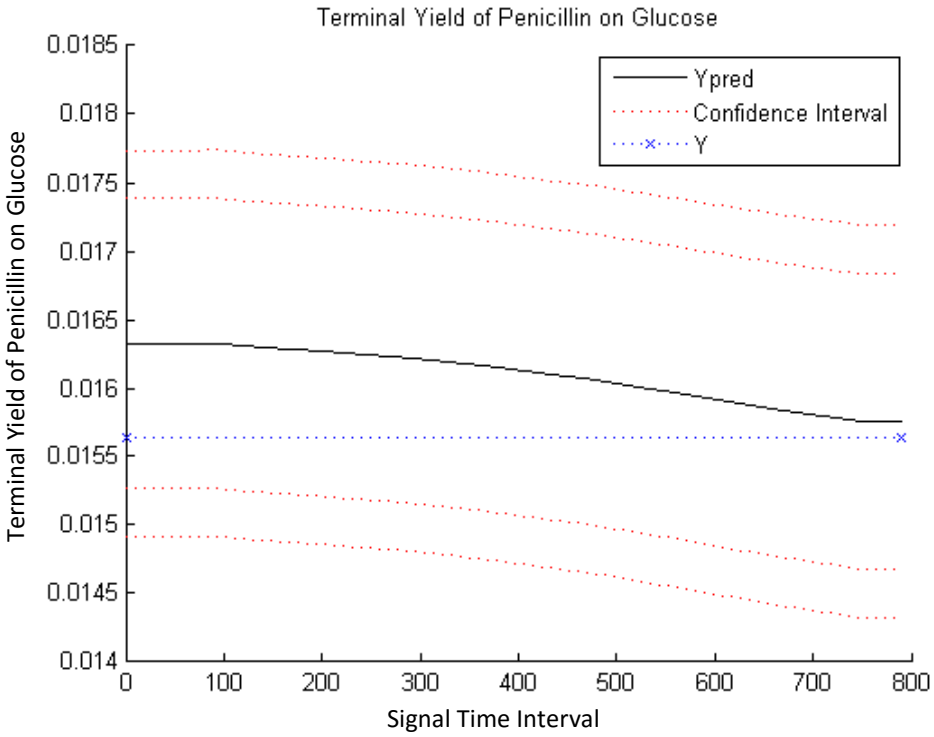


Figure C.15: Terminal Yield of Penicillin on Glucose (Global Model) – Case Study 2

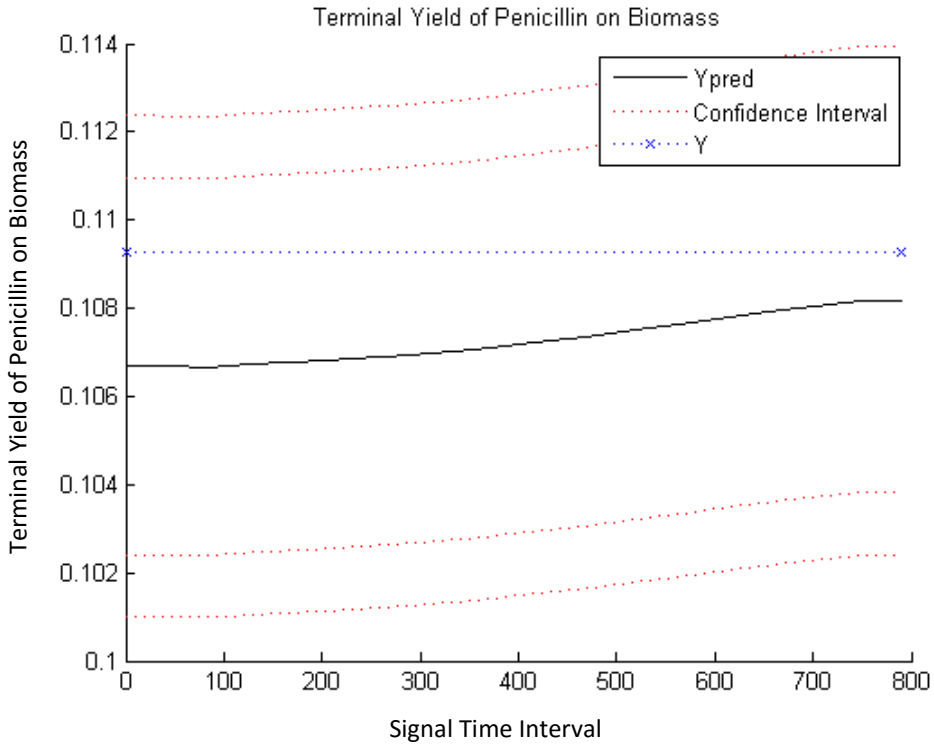


Figure C.16: Terminal Yield of Penicillin on Biomass (Global Model) – Case Study 2

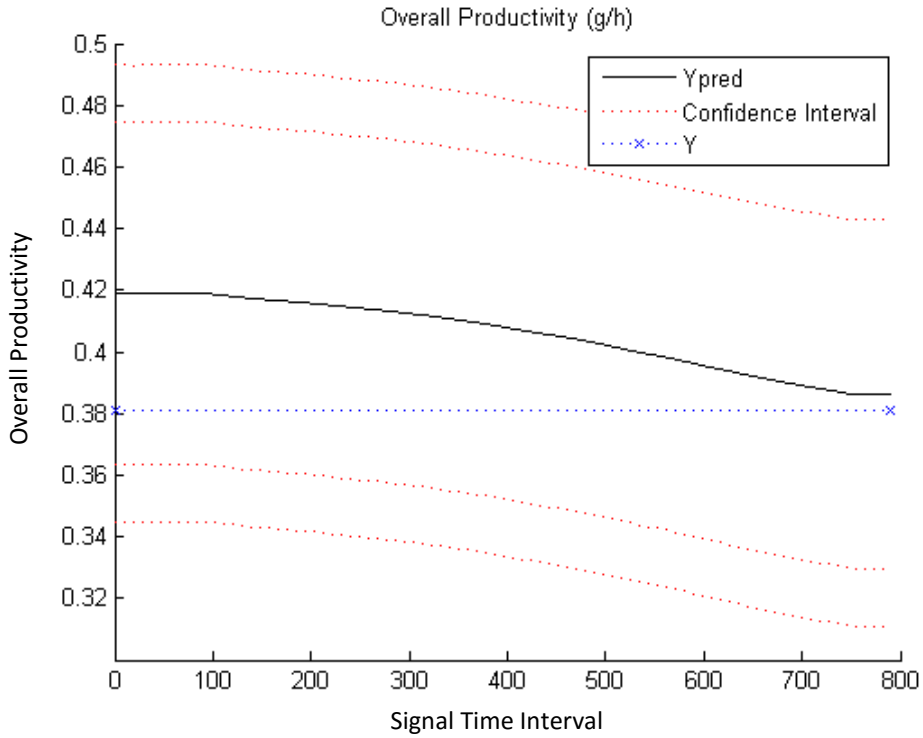


Figure C.17: Overall Productivity (Global Model) – Case Study 2

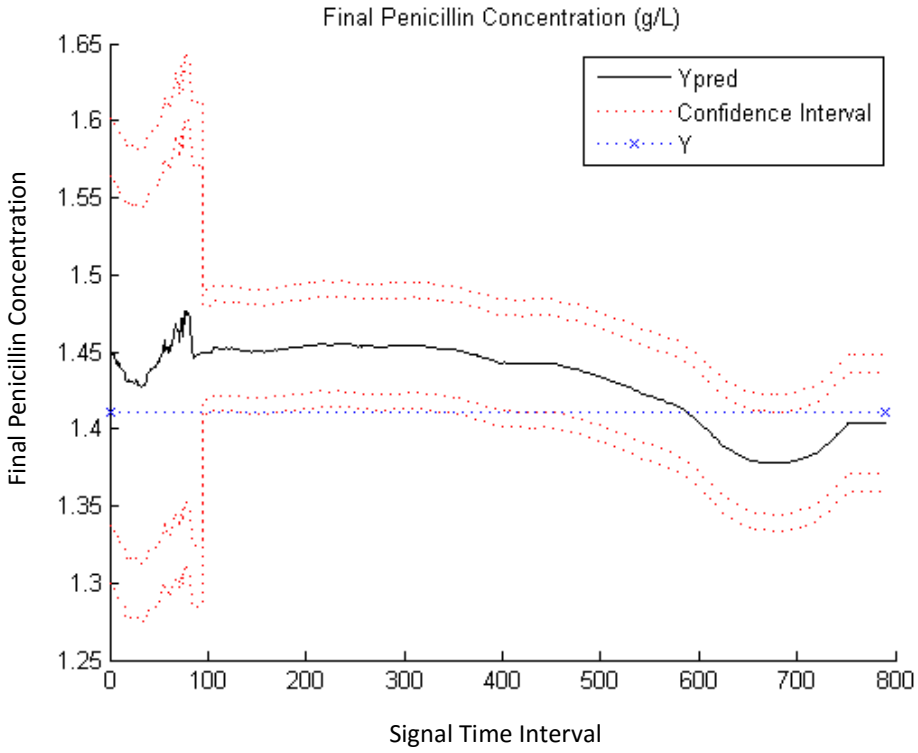


Figure C.18: Final Penicillin Concentration (Operational Phase Model) – Case Study 2

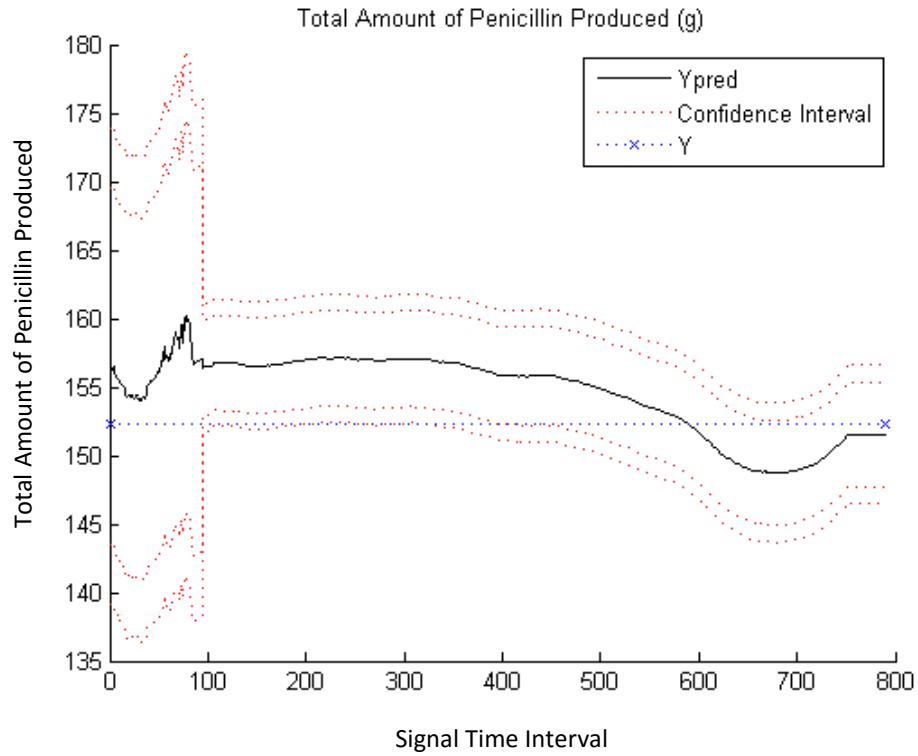


Figure C.19: Total Amount of Penicillin Produced (Operational Phase Model) – Case Study 2

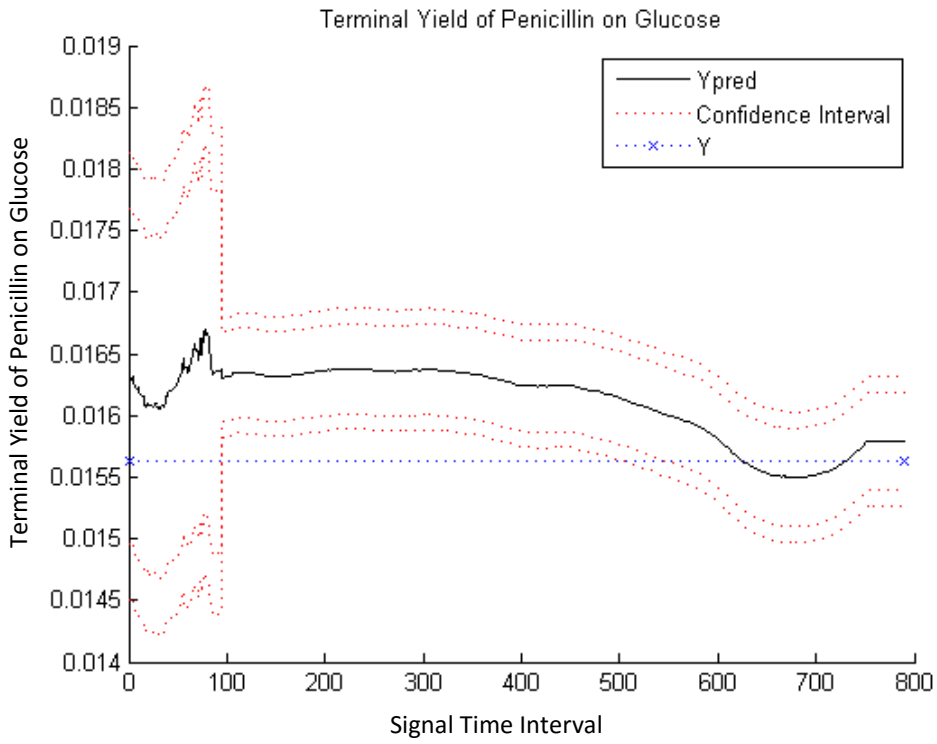


Figure C.20: Terminal Yield of Penicillin on Glucose (Operational Phase Model) – Case Study 2

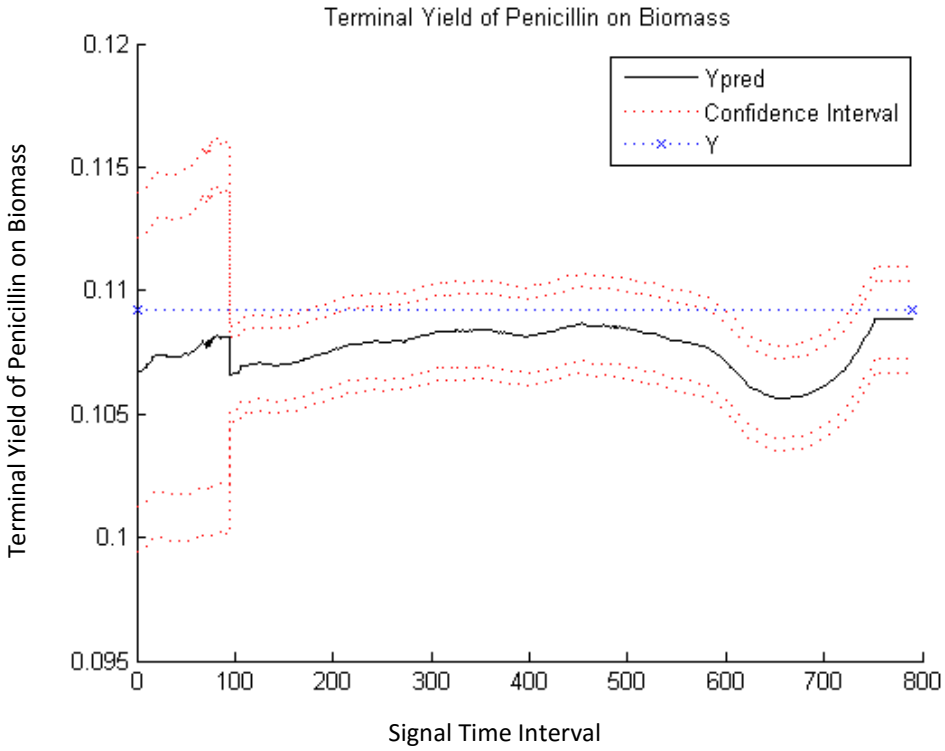


Figure C.21: Terminal Yield of Penicillin on Biomass (Operational Phase Model) – Case Study 2

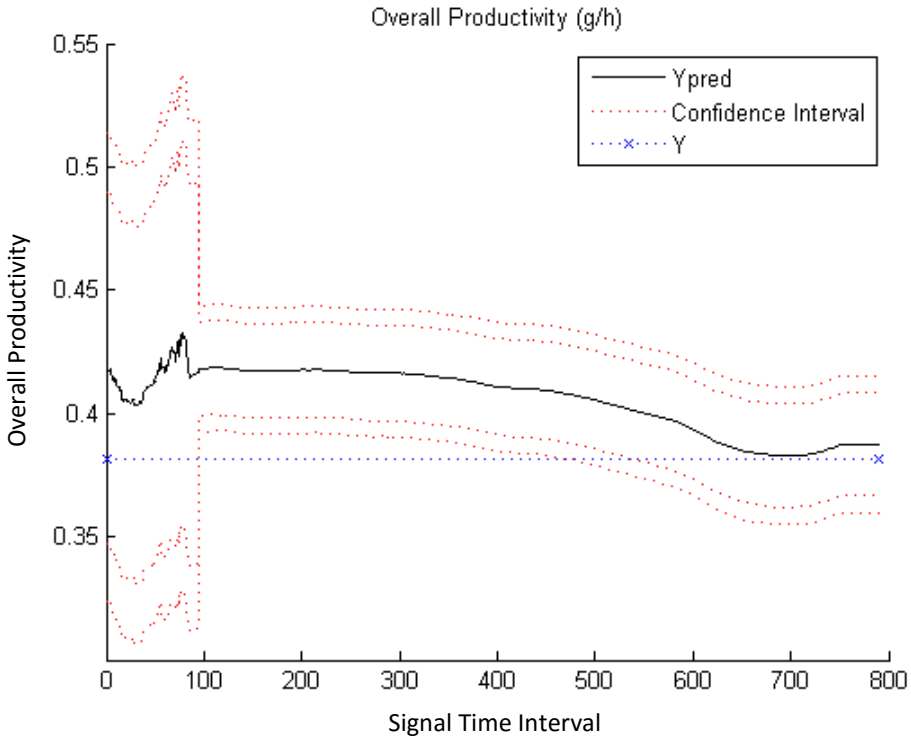


Figure C.22: Overall Productivity (Operational Phase Model) – Case Study 2

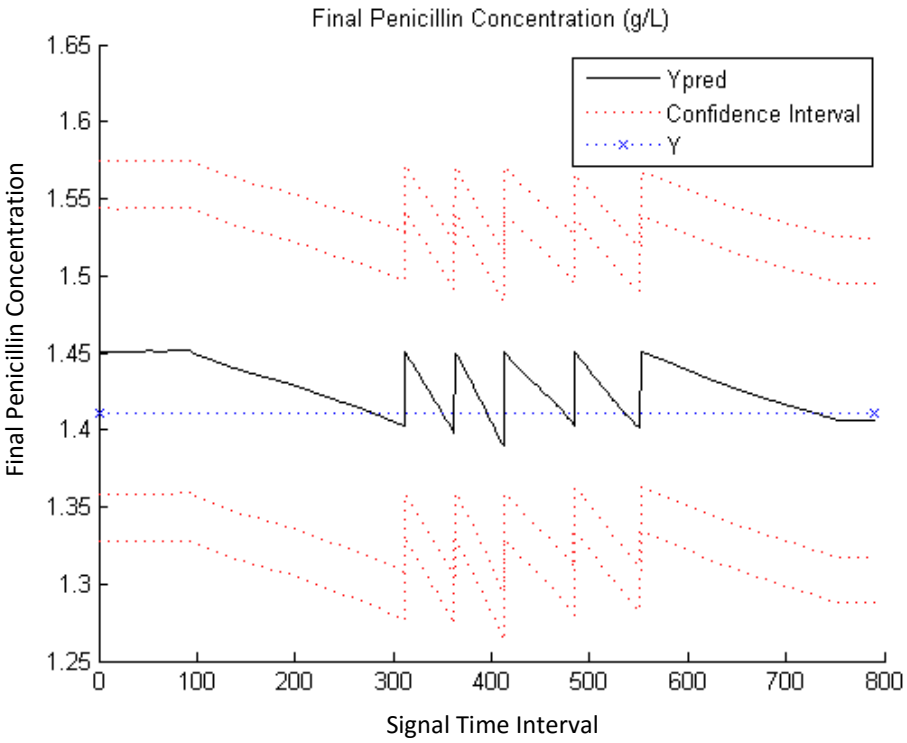


Figure C.23: Final Penicillin Concentration (Multiphase Model) – Case Study 2

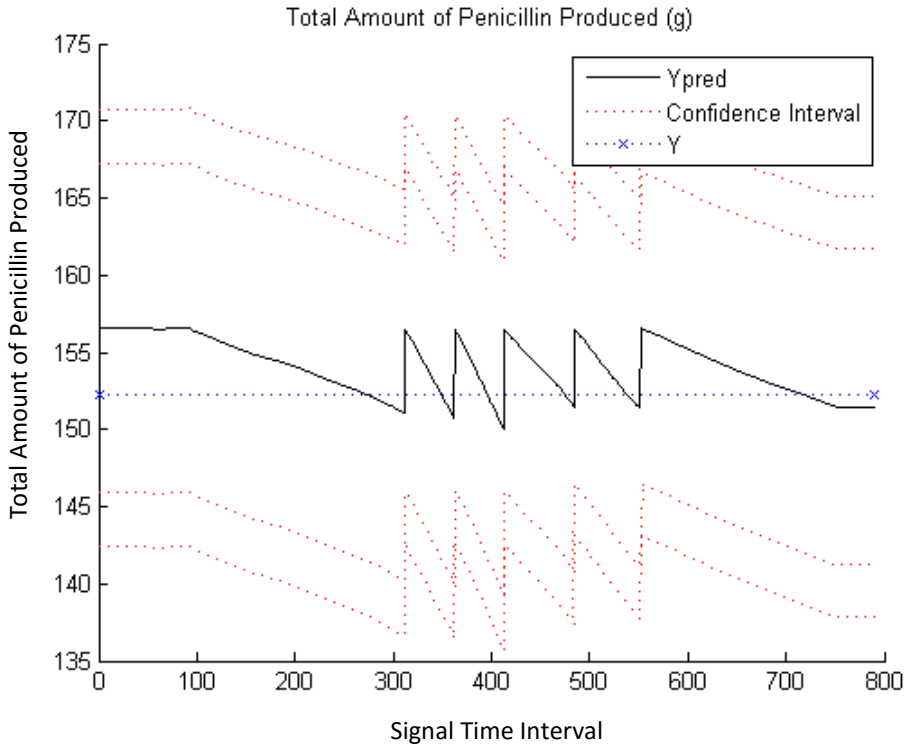


Figure C.24: Total Amount of Penicillin Produced (Multiphase Model) – Case Study 2

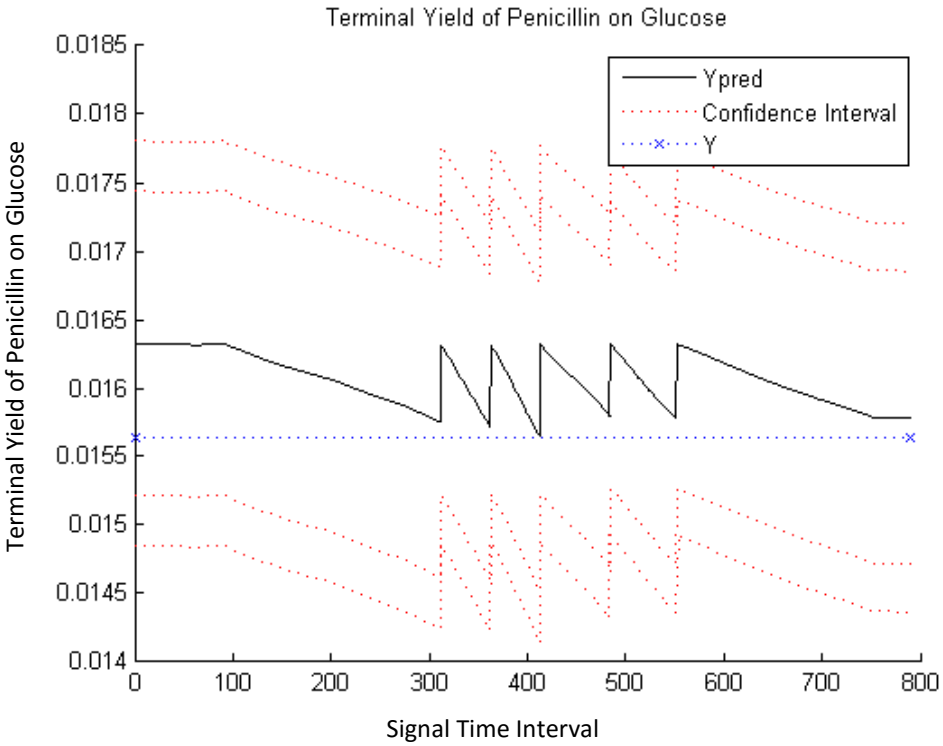


Figure C.25: Terminal Yield of Penicillin on Glucose (Multiphase Model) – Case Study 2

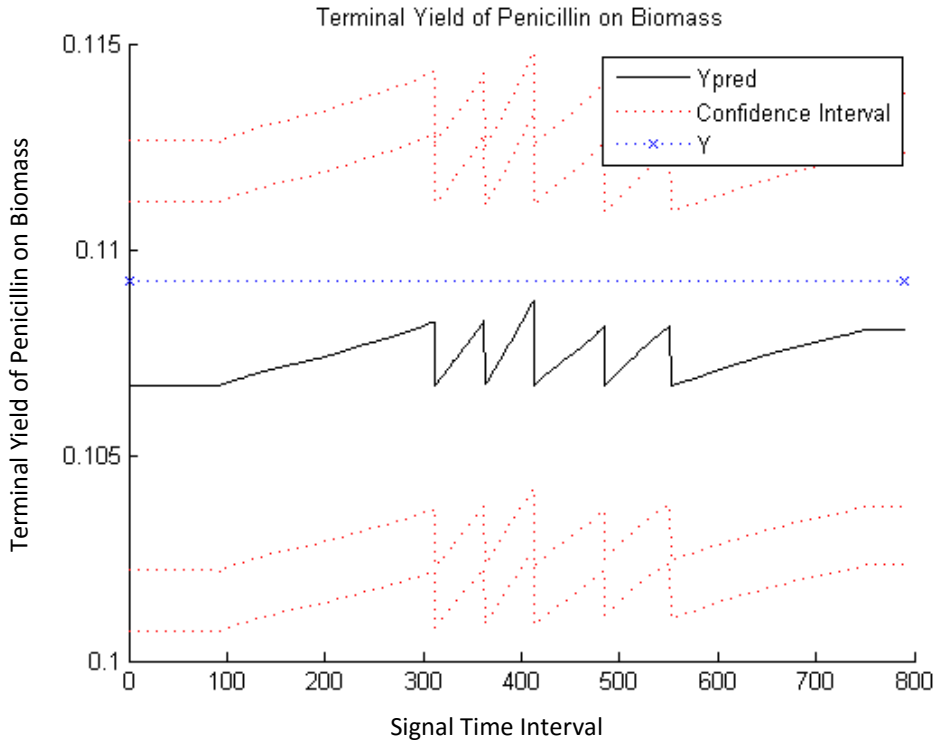


Figure C.26: Terminal Yield of Penicillin on Biomass (Multiphase Model) – Case Study 2

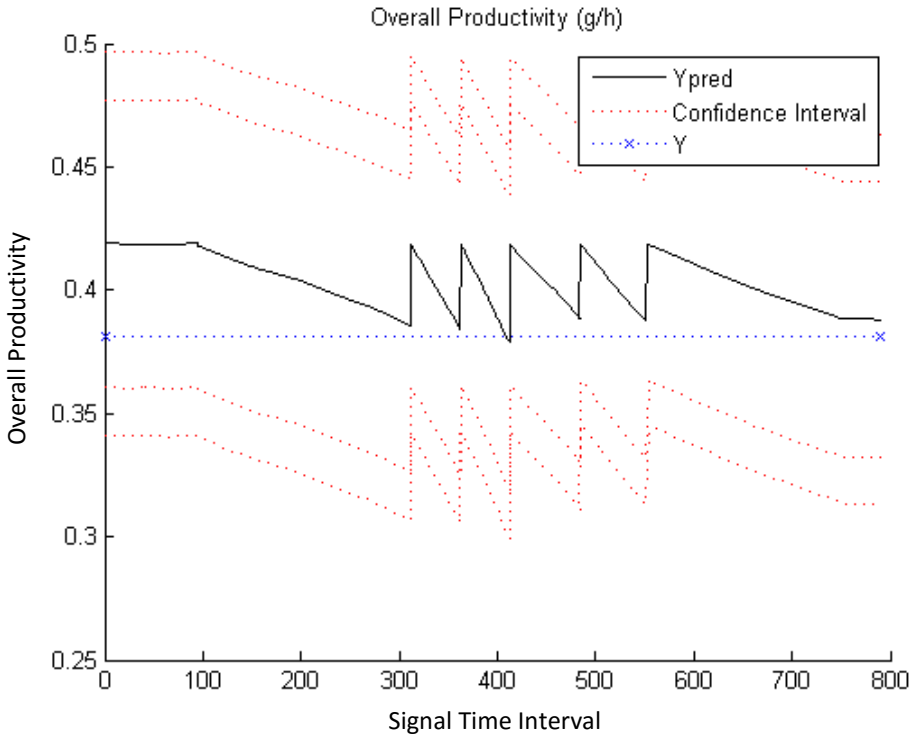


Figure C.27: Overall Productivity (Multiphase Model) – Case Study 2

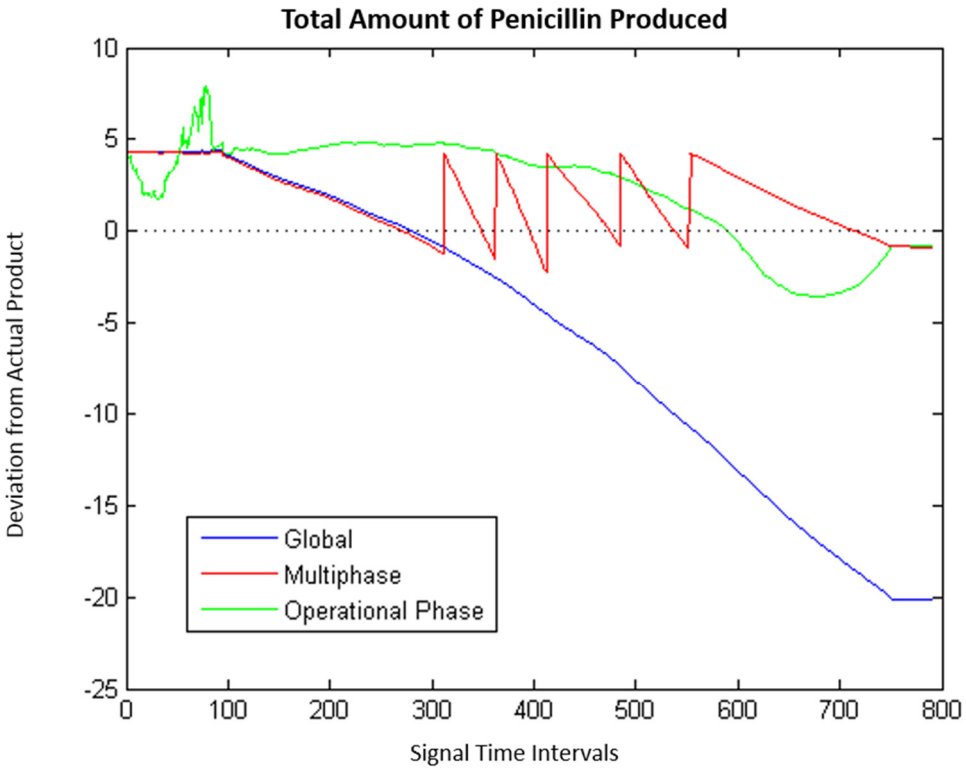
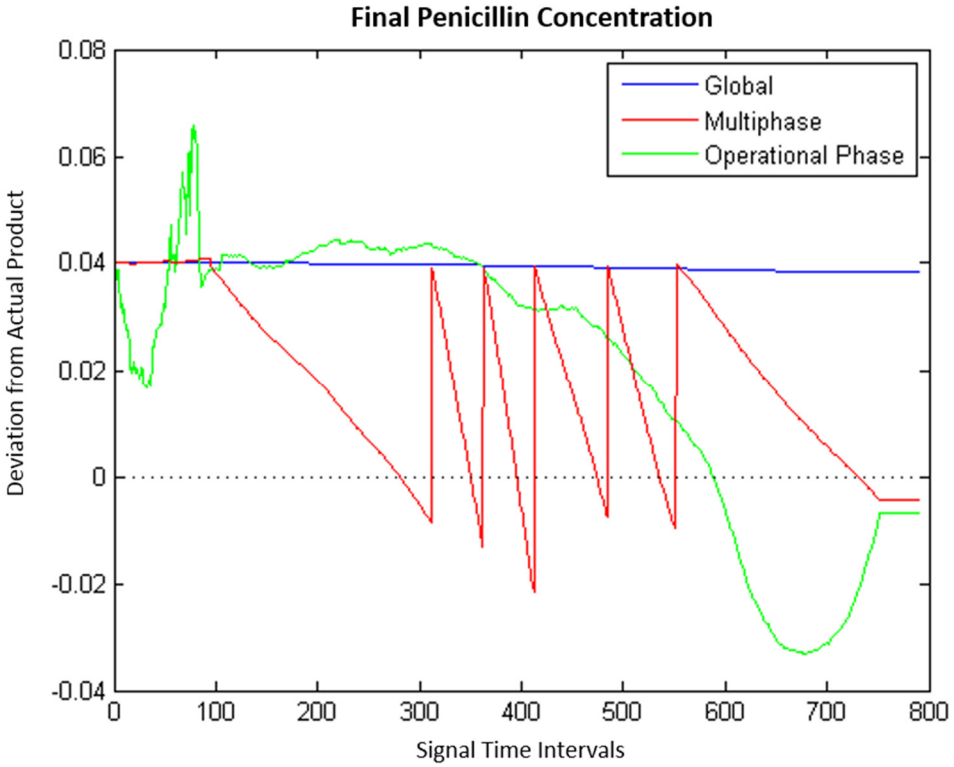


Figure C.28: Prediction Deviations from Actual Quality – Case Study 2

a)



b)

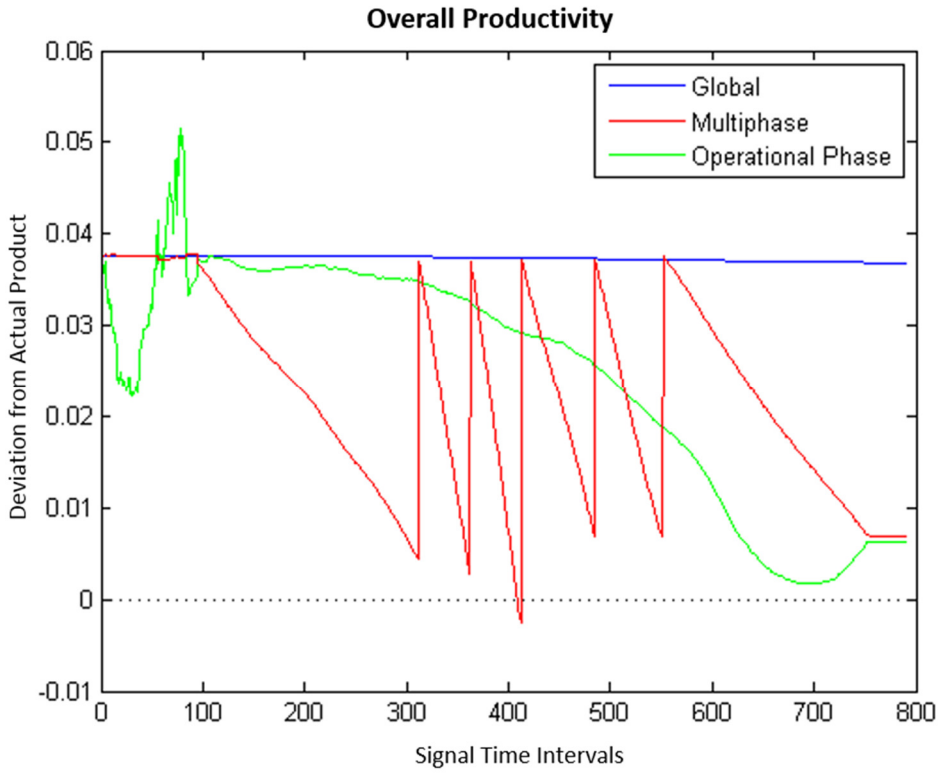
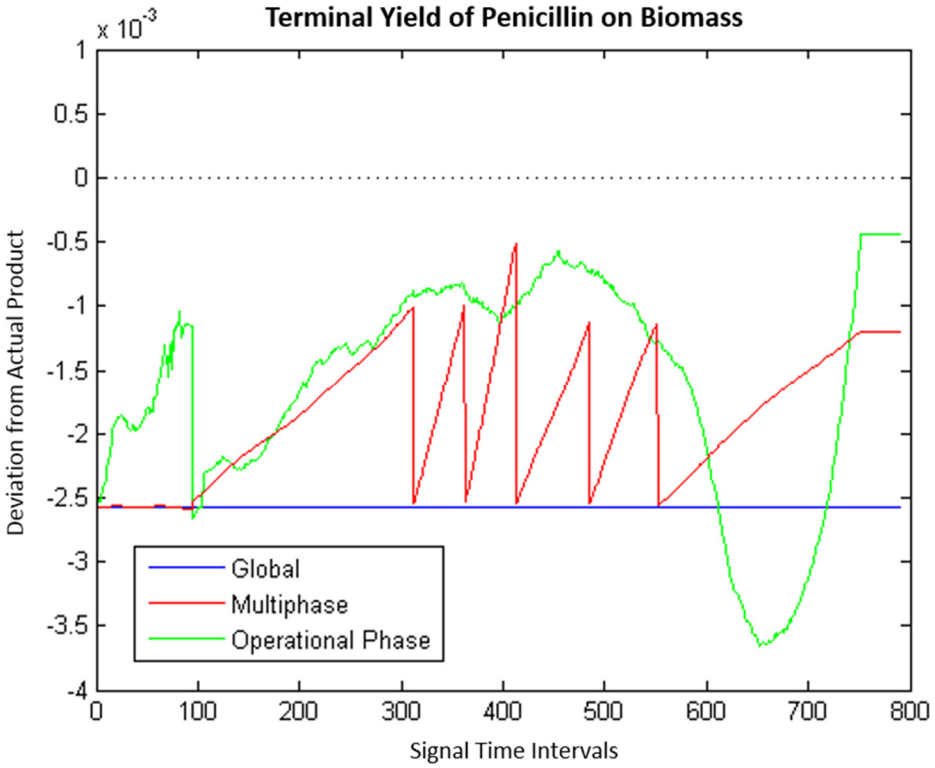


Figure C.29: Prediction Deviations from Actual Quality – Case Study 2

a)



b)

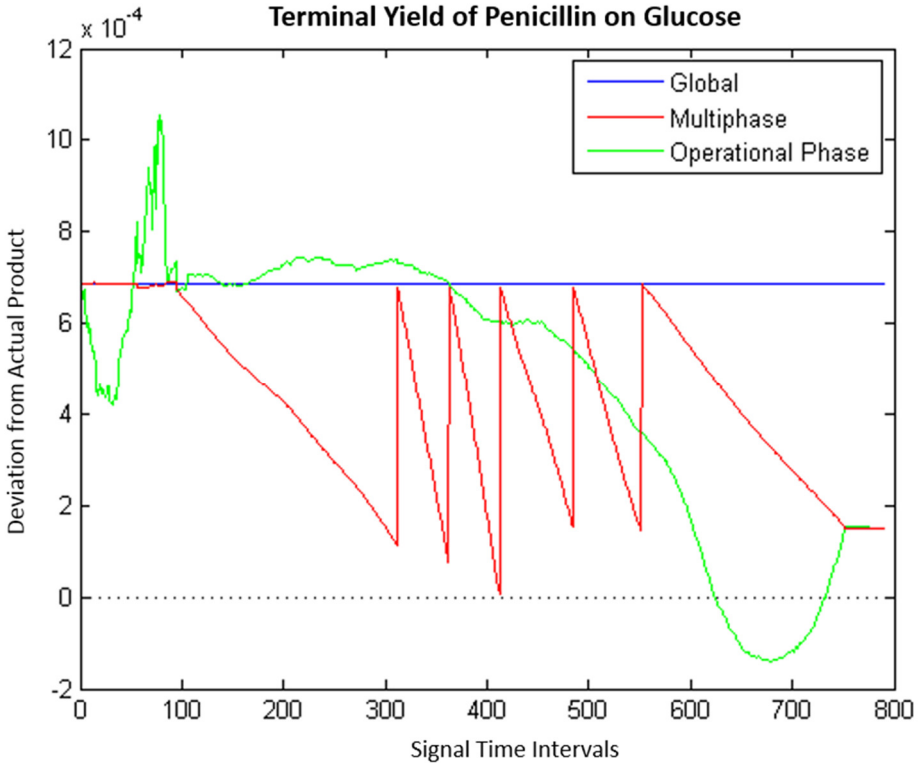


Figure C.30: Prediction Deviations from Actual Quality – Case Study 2

APPENDIX D: Final Concentrate Dissolution Modelling Results

Figure D.1 below shows the training of a global model and application thereof to the test data for the Final Concentrate Dissolution Process. 7 PCs were retained (r) accounting for 59% of the variance of the end-of-batch quality variables (y). The lines marked "test" in figure a) show the MSEs of the left out batches when using Leave-One-Out Cross Validation on the training batch and were used to assess the performance of the models.

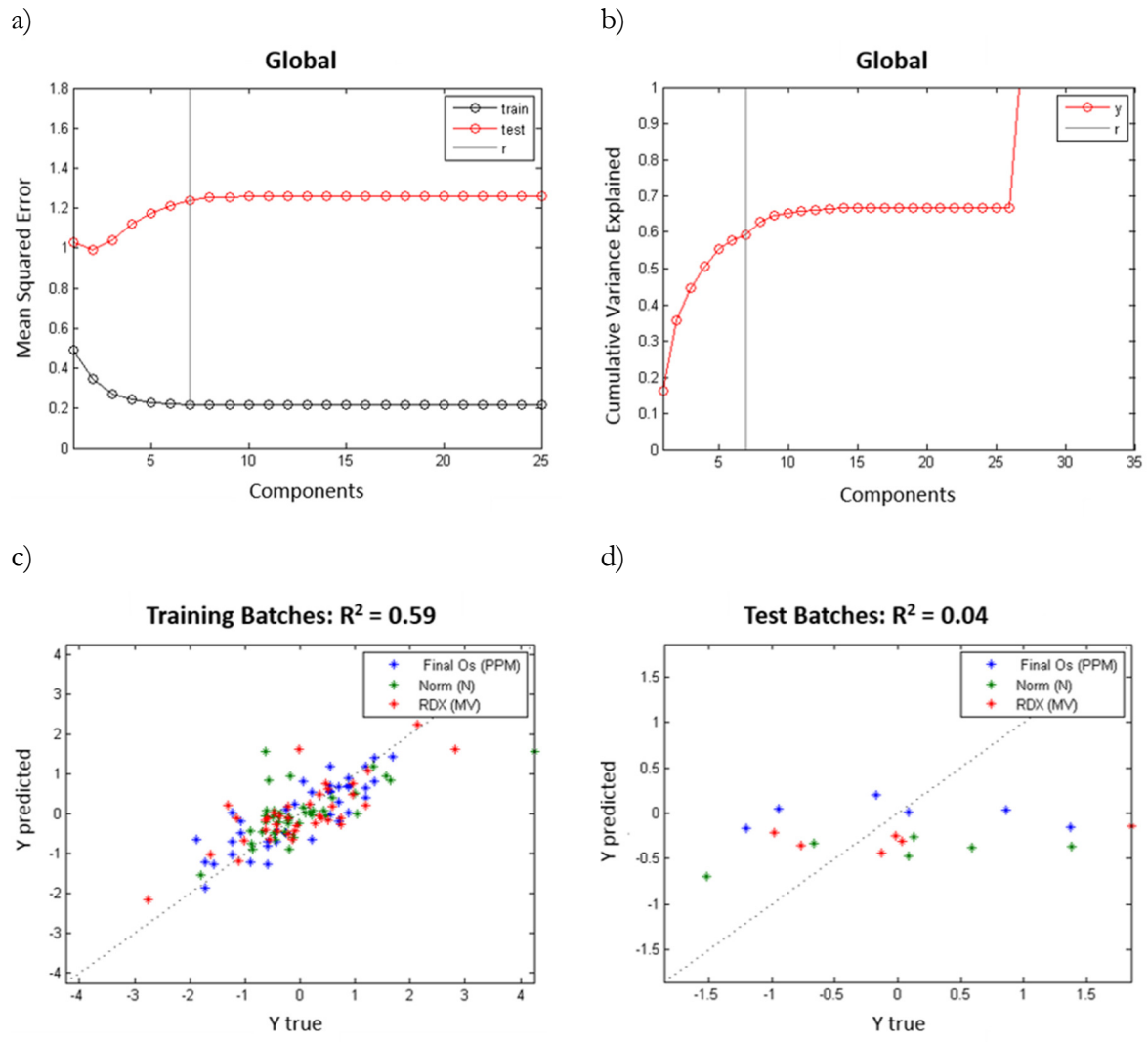


Figure D.1: Modelling results for global MPLS model

The application of the model to the test batches produced a very poor correlation. It is most likely that the model is over fitting, resulting in the large errors on the test data. Reducing the number of retained PCs would decrease the MSE, but the resulting model would explain an unsatisfactory proportion of the variance.

Figures D.2 a)-j) show the MSEs and cumulative variances in the end-of-batch quality that could be explained by MPLS modelling for each of the 7 operational phases of the Final Concentrate

Dissolution Process as a function of number of PCs retained. No model could explain more than 70% of the variance for any of the phases, and in most cases it required a large number of retained PCs (>15) to do so. In phases 1 – 4 and 6 the MSEs increased significantly as the number of PCs retained became larger, indicating poorer model accuracy. A compromise between minimising the MSEs and maximising the cumulative variance explained is shown by the vertical lines (r). Here, the amount of variance explained by the models was quite low for most of the phases, ranging from 37% variance explained (phase 3) to 62% variance explained (phase 5).

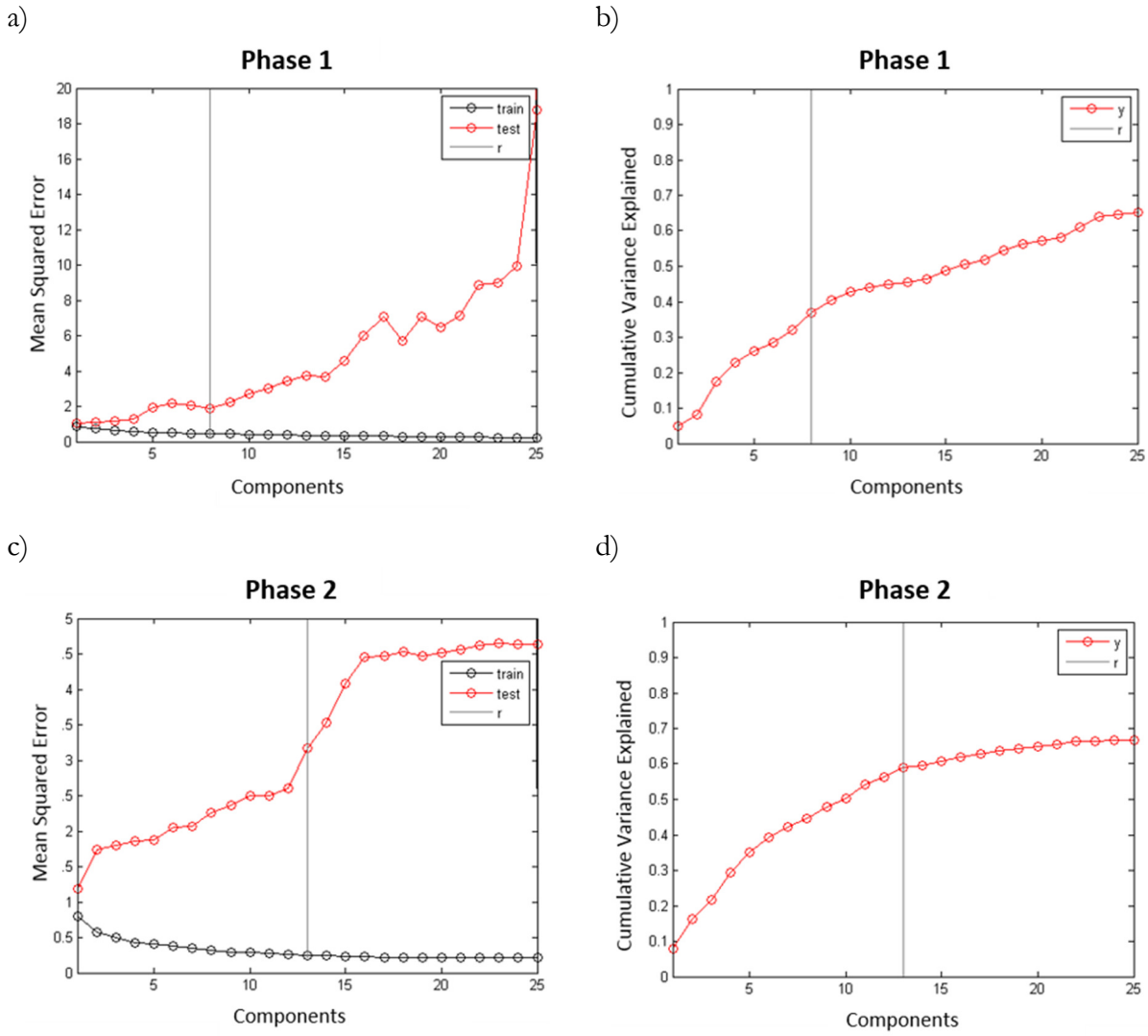
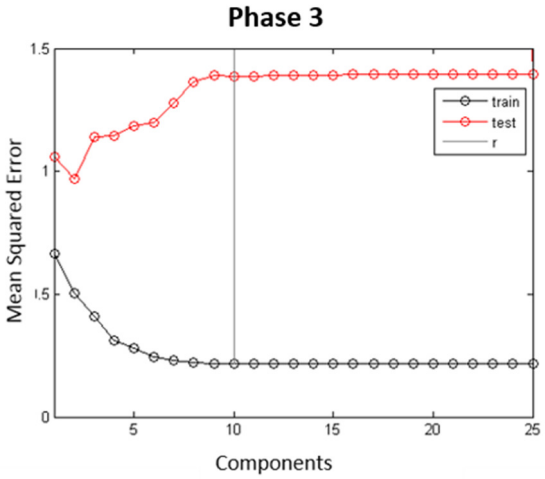
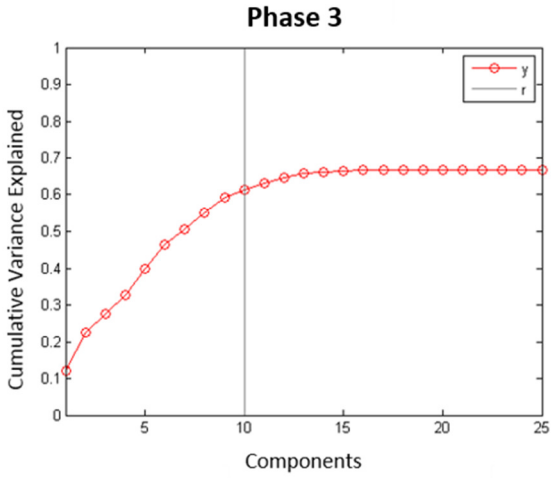


Figure D.2 (a-d): Modelling results for separate operational phase MPLS models

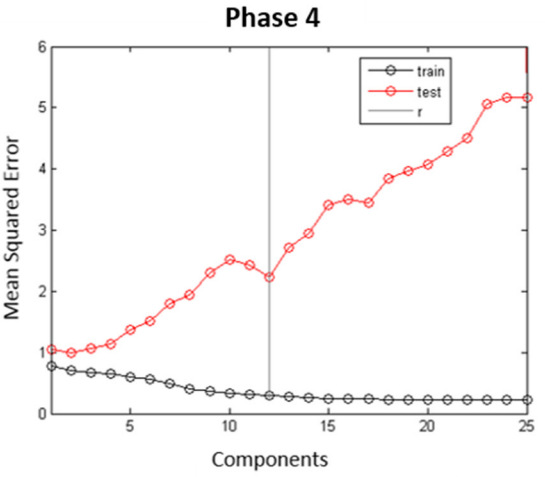
e)



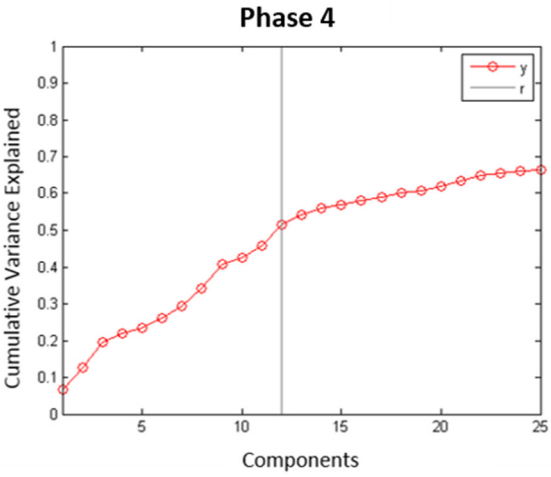
f)



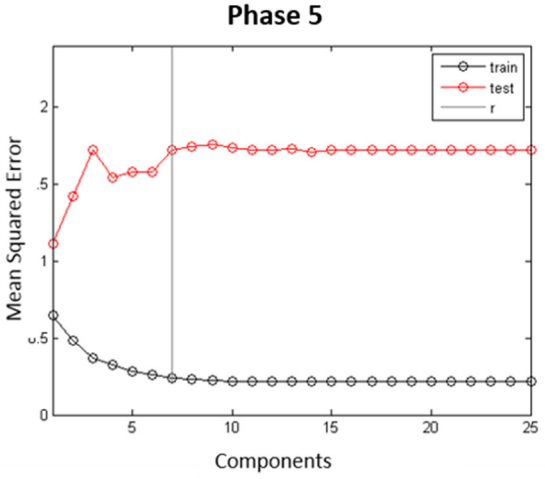
g)



h)



i)



j)

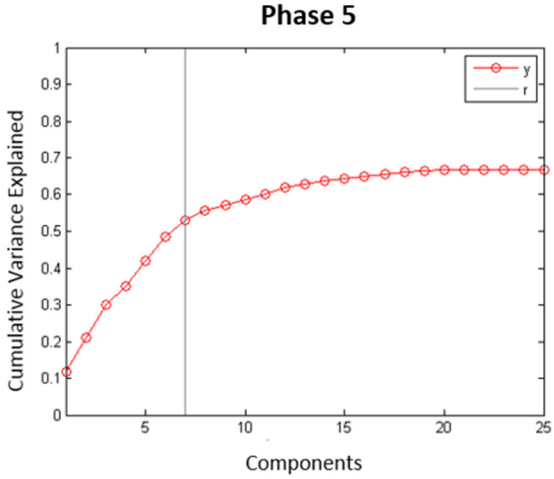


Figure D.2 (e-j): Modelling results for separate operational phase MPLS models

The analyses above show that the data could not be modelled appropriately to predict the end-of-batch quality using MPLS techniques. Training a global model resulted in over fitting and training separate operational phase models could not explain a satisfactory amount of variance in most of the phases. The conclusion was made that the data could not be used to create accurate models to predict the final quality, hence the modelling of this case study was excluded from analysis in the body of the thesis.

APPENDIX E: Hardware and Software

E.1 Hardware

Computations were performed on a desktop computer as well as through a server. The desktop computer had an Intel® Core™2 Quad CPU with 2.83 GHz processing speed and 4.00 GB (3.25 GB Usable) RAM. The server had an Intel® Zeon® CPU with 8 3.33 GHz processors and 32 GB RAM. Computational speed calculations were done using the latter computer.

E.2 Software

The operating systems of the two computers used were:

- Microsoft Windows (32-bit) 7 Enterprise (desktop)
- Microsoft Windows (64-bit) Server 2008 R2 Enterprise (server)

Analyses were done using MATLAB R2014a. Simulink was used to create a the three-tank system simulation (chapter 4) and PenSim v2.0 files were obtained and used to generate data for the penicillin cultivation process (chapter 5).

APPENDIX F: Detailed Description of Algorithms Used

F.1 Off-line Training of Data

Models were trained off-line from NOC data. The NOC data consisted of a number of batches, I. For each batch, measurements were recorded for a number of variables J at each time step k_i for K_i time steps. These data were then arranged in a $K_i \times J$ matrix. The starting and end time steps for each user-defined phase p were also known. All batches had the same number of phases. The data were pre-processed before the models could be trained using the pre-processing routine.

F.1.1 Pre-processing Routine

Given: I batches of NOC data, each of size $K_i \times J$

Corresponding set of Π phase boundary indexes for each batch i

Phase division method

Step 1 – Preliminary Phase Division

For each phase π, isolate data from the starting phase boundary to the end phase boundary.

Result: Π sets of I batches, each of size $K_{\pi,i} \times J$.

Step 2 – Synchronisation

For each phase π

1. Slab scale the data:
    - Concatenate all the batches such that the resulting matrix is of size $J \times \sum_{i=1}^{I} K_{\pi,i}$
    - Find the mean $\mu_{\pi,slab,j}$ for each of the J columns
    - Find the standard deviation $s_{\pi,slab,j}$ for each of the J columns. If the standard deviation of any of the columns is equal to zero, set equal to 1
    - For each batch at every time step k, subtract the mean of each of the J columns from its corresponding variable ($\tilde{x}_{\pi,raw,ijk}$) and divide by its corresponding standard deviation:

$$x_{\pi,slab,ijk} = \frac{\tilde{x}_{\pi,raw,ijk} - \mu_{\pi,slab,j}}{s_{\pi,slab,j}} \qquad \text{F.1}$$

- Store each $x_{\pi,slab,ijk}$ in $\boldsymbol{X_{\pi,slab,i}} \in \mathbb{R}^{K_{\pi,i} \times J}$ for each batch *i*, as well as $\mu_{\pi,slab,j}$ and $s_{\pi,slab,j}$ for each variable *j* in $\boldsymbol{\mu_{\pi,slab}} \in \mathbb{R}^{1 \times J}$ and $\boldsymbol{s_{\pi,slab}} \in \mathbb{R}^{1 \times J}$ respectively

2. Synchronise the data
   - Select an initial, temporary reference batch $\boldsymbol{X_{tempref}} \in \mathbb{R}^{K_{\pi,tempref} \times J}$ from the training batches with the number of time intervals closest to the average number of time intervals across all the batches.
   - Synchronise each $\boldsymbol{X_{slab,i}}$ to $\boldsymbol{X_{tempref}}$ using the synchronisation routine (section F.1.3). The dimensions of the synchronised matrix for each *i*, $\boldsymbol{X_{tempsync,i}} \in \mathbb{R}^{K_{\pi,tempref} \times J}$ will be the same.
   - Select a reference batch $\boldsymbol{X_{ref}} \in \mathbb{R}^{K_{\pi,ref} \times J}$ from the *I* batches as the batch with the maximum cumulative product of correlation coefficients with all the other batches. The correlation coefficient r_{ab} between two batches $\boldsymbol{X_a}$ and $\boldsymbol{X_b}$ is given by:

$$r_{ab} = \left(\frac{\boldsymbol{X_a} \times \boldsymbol{X_b}^{\boldsymbol{T}}}{norm(\boldsymbol{X_a}) \times norm(\boldsymbol{X_b})} \right)^{\mathbf{2}} \qquad \text{F.2}$$

   - Synchronise each $\boldsymbol{X_{tempsync,i}}$ to $\boldsymbol{X_{ref}}$ using the synchronisation routine. Each resulting synchronised matrix $\boldsymbol{X_{sync,i}} \in \mathbb{R}^{K_{\pi,ref} \times J}$ will have the same dimensions.

3. Unscale the synchronised data
   - Recall the means $(\mu_{\pi,slab,j})$ and standard deviations $(s_{\pi,slab,j})$ calculated in step 1.
   - For each batch at every reference batch time step *k*, multiply the standard deviation of each of the *J* columns with its corresponding variable $(x_{sync,ijk})$ and add its corresponding mean:

$$\tilde{x}_{\pi,sync,ijk} = \left(x_{\pi,sync,ijk} \times s_{\pi,slab,j}\right) + \mu_{\pi,slab,j} \qquad \text{F.3}$$

   - The matrix for each batch $\tilde{\boldsymbol{X}}_{\boldsymbol{\pi,sync,i}} \in \mathbb{R}^{K_{\pi,ref} \times J}$ contains these values.

Step 3 – Phase Division

1. If the phase division method is set to "none", the data are unchanged and $P = \Pi$

2. Otherwise, for each batch *i*, concatenate the data for each phase such that $\tilde{\boldsymbol{X}}_{\boldsymbol{sync,i}} \in \mathbb{R}^{K_{ref} \times J}$, where K_{ref} is the number of time intervals of the entire reference batch:

$$K_{ref} = \sum_{\Pi}^{\pi=1} K_{\pi,ref} \qquad \text{F.4}$$

3. If the phase division method is set to "global" the data are unchanged and $P = 1$
4. If the phase division method is set to "multiphase", execute the multiphase division routine (section 7.1.4)
    - Overwrite the existing phase boundaries with the new phase boundaries
    - Repeat step 1 with the new phase boundaries

Step 4 – Auto Scaling

For each phase *p*

1. Concatenate all the batches such that the resulting matrix is of size $K_{p,ref} \times (J \times I)$
2. Reshape the matrix to size $I \times (K_{p,ref} \times J)$
3. Find the mean $\mu_{p,auto,jk}$ for each of the $K_{p,ref} \times J$ columns
4. Find the standard deviation $s_{p,auto,jk}$ for each of the $K_{p,ref} \times J$ columns. If the standard deviation of any of the columns is equal to zero, set equal to 1
5. For each batch, subtract the mean of each of the $K_{p,ref} \times J$ columns from its corresponding variable ($x_{p,sync,ijk}$) and divide by its corresponding standard deviation:

$$x_{p,auto,ijk} = \frac{\tilde{x}_{p,sync,ijk} - \mu_{p,auto,jk}}{s_{p,auto,jk}} \qquad \text{F.5}$$

6. Store each $x_{p,auto,ijk}$ in $\boldsymbol{X}_{\boldsymbol{p,auto,i}} \in \mathbb{R}^{K_{p,ref} \times J}$ for each batch *i*, as well as $\mu_{p,auto,jk}$ and $s_{p,auto,jk}$ for each variable *j* and time index *k* in $\boldsymbol{\mu}_{\boldsymbol{p,auto}} \in \mathbb{R}^{K_{p,ref} \times J}$ and $\boldsymbol{s}_{\boldsymbol{p,auto}} \in \mathbb{R}^{K_{p,ref} \times J}$ respectively

Step 5 – Batch-wise Unfolding

For each phase *p*

1. Concatenate all the batches $\boldsymbol{X}_{\boldsymbol{p,auto,i}}$ such that the resulting matrix is of size $J \times (K_{p,ref} \times I)$
2. Reshape the matrix to size $I \times (JK_{p,ref})$. The resulting matrix $\boldsymbol{X}_{\boldsymbol{p,BW}} \in \mathbb{R}^{I \times (JK_{p,ref})}$

F.1.2 Model Training Routine

Given: *P* sets synchronised, auto-scaled and unfolded matrices, $\boldsymbol{X_{p,BW}} \in \mathbb{R}^{I \times (JK_{p,ref})}$

I batches of product data with *M* product variables, $\widetilde{\boldsymbol{Y}} \in \mathbb{R}^{I \times M}$

A sets of *P* phase boundary indexes

A vector of confidence thresholds

Phase Division method

Step 1 – Product data scaling

1. Find the mean $\mu_{prod,m}$ for each of the M columns
2. Find the standard deviation $s_{prod,m}$ for each of the M columns. If the standard deviation of any of the columns is equal to zero, set equal to 1
3. For each batch, subtract the mean of each of the M columns from its corresponding variable (y_m) and divide by its corresponding standard deviation:

$$y_m = \frac{\tilde{y}_m - \mu_{prod,m}}{s_{prod,m}} \qquad \text{F.6}$$

4. Store each y_m in $\boldsymbol{Y} \in \mathbb{R}^{I \times M}$ for each batch *i*, as well as $\mu_{prod,m}$ and $s_{prod,m}$ for each product variable *m* in $\boldsymbol{\mu_{prod}} \in \mathbb{R}^{I \times M}$ and $\boldsymbol{s_{prod}} \in \mathbb{R}^{I \times M}$ respectively

Step 2 – Select number of components to retain

For each phase *p*

1. If the phase division method is set to "multiphase", recall the number of retained components A
2. Otherwise, perform Leave One Out Cross Validation (LOOCV). For $A = 1{:}\,I - 4$
   - For every batch *i*
     - Isolate batch *i* such that $\boldsymbol{x_{i,unfold}} \in \mathbb{R}^{1 \times (JK_{p,ref})}$
     - Remove batch *i* from the unfolded matrix such that $\boldsymbol{X^*_{BW}} \in \mathbb{R}^{(I-1) \times (JK_{p,ref})}$
     - Use the SIMPLS algorithm to find the regression matrix $\boldsymbol{\beta} \in \mathbb{R}^{(JK_{p,ref}) \times M}$
     - Compute $\hat{\boldsymbol{y}}_{\boldsymbol{i}} \in \mathbb{R}^{1 \times M}$ from $\boldsymbol{x_{i,BW}}$ and $\boldsymbol{\beta}$ using equation 2.26

- Find the squared prediction error for every m

$$SPE_{y_{i,m}} = \left(y_{i,m} - \hat{y}_{i,m}\right)^2 \quad \text{F.7}$$

- Find the mean SPE_{y_i} across all m and store in the A-th entry the vector $\boldsymbol{SPE_y} \in \mathbb{R}^{1\times(I-4)}$
- Compute $\widehat{\boldsymbol{Y}}^* \in \mathbb{R}^{(I-1)\times M}$ from $\boldsymbol{X^*_{BW}}$ and $\boldsymbol{\beta}$ using equation 2.26
- Find the squared prediction error for every m and every i

$$SPE_{Y^*_{i,m}} = \left(Y^*_{i,m} - \hat{Y}^*_{i,m}\right)^2 \quad \text{F.8}$$

- Find the mean $SPE_{Y^*_m}$ across all i
- Find the mean SPE_{Y^*} across all m and store in the A-th entry in the vector $\boldsymbol{SPE_{Y^*}} \in \mathbb{R}^{1\times(I-4)}$

- Plot $\boldsymbol{SPE_y}$ and $\boldsymbol{SPE_{Y^*}}$ on a graph
- Manually select the desired number of retained components via inspection of the trends on the graph

Step 3 – Train the model

For each phase p

- Use the SIMPLS algorithm and the number of retained components chosen to train a PLS model from $\boldsymbol{X_{p,unfold}}$ and $\boldsymbol{Y}$
- Store the x-loadings matrix $\boldsymbol{P_p} \in \mathbb{R}^{A\times(JK_{p,ref})}$, the x-scores matrix $\boldsymbol{T_p} \in \mathbb{R}^{I\times A}$, the regression coefficients $\boldsymbol{\beta_p} \in \mathbb{R}^{(JK_{p,ref})\times M}$, the weights matrix $\boldsymbol{R_p} \in \mathbb{R}^{(JK_{p,ref})\times A}$ from the result
- Compute $\widehat{\boldsymbol{Y}} \in \mathbb{R}^{I\times M}$ from $\boldsymbol{X_{p,unfold}}$ and $\boldsymbol{\beta_p}$ using equation 2.26.
- Subtract $\boldsymbol{Y}$ from $\widehat{\boldsymbol{Y}}$ and square each element of the resulting matrix
- Calculate the mean of each row, $\boldsymbol{MSE} \in \mathbb{R}^{I\times 1}$
- Calculate the mean of $\boldsymbol{MSE}$ to and store as MSE_y

Step 4 – Calculate the Monitoring Statistic Parameters

For each phase p

1. For every k
   - Isolate the 1st to the kJ-th columns of the $\boldsymbol{X_{p,unfold}}$ matrix to create $\boldsymbol{X_{p,k}} \in \mathbb{R}^{I \times (Jk)}$
   - Pad $\boldsymbol{X_{p,k}}$ with zeros up until time $K_{p,ref}$ such that $\boldsymbol{X_{p,k}} \in \mathbb{R}^{I \times (JK_{p,ref})}$
   - Isolate the 1st to the kJ-th rows of the $\boldsymbol{R_p}$ matrix to create $\boldsymbol{R_{p,k}} \in \mathbb{R}^{(Jk) \times A}$
   - Pad $\boldsymbol{R_k}$ with zeros up until time $K_{p,ref}$ such that $\boldsymbol{R_{p,k}} \in \mathbb{R}^{(JK_{p,ref}) \times A}$
   - Calculate $\boldsymbol{T_{p,k}} \in \mathbb{R}^{I \times A}$ from

$$\boldsymbol{T_{p,k}} = \boldsymbol{X_{p,k}} \cdot \boldsymbol{R_{p,k}} \qquad \text{F.9}$$

   - Calculate the variances of each column and store in $\boldsymbol{s_{p,k}} \in \mathbb{R}^{1 \times A}$
   - Store $\boldsymbol{s_{p,k}}$ in the k-th row of the matrix $\boldsymbol{S_p}$
   - Calculate $\widehat{\boldsymbol{X}}_{\boldsymbol{p,k}} \in \mathbb{R}^{I \times (JK_{p,ref})}$ from

$$\widehat{\boldsymbol{X}}_{\boldsymbol{p,k}} = \boldsymbol{T_{p,k}} \cdot \boldsymbol{P_{p,k}} \qquad \text{F.10}$$

   - Reshape $\boldsymbol{X_{p,k}}$ and $\widehat{\boldsymbol{X}}_{\boldsymbol{p,k}}$ to $\underline{\boldsymbol{X}}_{\boldsymbol{p,k}} \in \mathbb{R}^{I \times J \times K_{p,ref}}$ and $\underline{\widehat{\boldsymbol{X}}}_{\boldsymbol{p,k}} \in \mathbb{R}^{I \times J \times K_{p,ref}}$
   - Subtract $\underline{\widehat{\boldsymbol{X}}}_{\boldsymbol{p,k}}$ from $\underline{\boldsymbol{X}}_{\boldsymbol{p,k}}$ and square each element of the resulting matrix to get $\underline{\boldsymbol{SPE}}_{\boldsymbol{p,k}} \in \mathbb{R}^{I \times J \times K_{p,ref}}$
   - For every batch i
     - Isolate the i-th batch $\boldsymbol{SPE_{p,k,i}} \in \mathbb{R}^{J \times K_{p,ref}}$
     - Isolate the k-th column such that $\boldsymbol{SPE_{p,k,i}} \in \mathbb{R}^{J \times 1}$
     - Find the sum of all the elements in $\boldsymbol{SPE_{p,k,i}}$ to get $SPE_{p,k,i}$
     - Store $SPE_{p,k,i}$ in the i-th column of the vector $\boldsymbol{SPE_{p,k}} \in \mathbb{R}^{1 \times I}$
   - Find the sum of all the elements in $\boldsymbol{SPE_{p,k}}$ to get $SPE_{p,k}$
   - Store $SPE_{p,k}$ as the k-th element in the vector $\boldsymbol{SPE_p} \in \mathbb{R}^{1 \times K_{p,ref}}$
   - Find the mean, $\mu_{p,k,SPE}$ and variance, $s_{p,k,SPE}$ of the elements in $\boldsymbol{SPE_p}$ from the $(k-1)$ to the $(k+1)$-th interval
   - Calculate $g_{p,k}$ from

$$g_{p,k} = \frac{s_{p,k,SPE}}{2\,\mu_{p,k,SPE}} \qquad \text{F.11}$$

- Calculate $g_{p,k}$ from

$$h_{p,k} = \frac{2\left(\mu_{p,k,SPE}\right)^2}{s_{p,k,SPE}} \qquad \text{F.12}$$

- Find the Normal variable z_α at significance values $\alpha = 0.05$ and 0.01
- Calculate the SPE limits for each α using the approximation for the Chi-squared variable

$$SPE_{p,k,\alpha} = g_{p,k}h_{p,k}\left[1 - \frac{2}{9h_{p,k}} + z_\alpha\left(\frac{2}{9h_{p,k}}\right)^{1/2}\right]^3 \qquad \text{F.13}$$

- Store each $SPE_{p,k,\alpha}$ as the k-th element in the vector $\boldsymbol{SPE_{p,\alpha}} \in \mathbb{R}^{1 \times K_{p,ref}}$

2. Calculate and store the Hotellings T^2 monitoring constant c_p from

$$c_p = \frac{I \cdot (I - A)}{A \cdot (I^2 - 1)} \qquad \text{F.14}$$

3. Find the Hotellings T^2 limits as the critical value of the F-statistic at significance values $\alpha = 0.05$ for A and A-I degrees of freedom and store as $F_{\alpha,A,I-A}$

F.1.3 Synchronisation Routine

Given: A signal batch of slab scaled data, $\boldsymbol{X_{sig}} \in \mathbb{R}^{K_{\pi,sig} \times J}$

A reference batch of slab scaled data, $\boldsymbol{X_{ref}} \in \mathbb{R}^{K_{\pi,ref} \times J}$

A matrix $\boldsymbol{W}$ of variable weights

The synchronisation method employed

Step 1 – Create a synchronisation grid

1. For each reference index k_{ref} from 1: $K_{\pi,ref}$, assign a set of signal indexes from 1: $K_{\pi,sig}$. The resulting grid is of size $K_{\pi,ref} \times K_{\pi,sig}$

Step 2 – Create feasibility map based on global constraints

1. For each reference index k_{ref} on the synchronisation grid, include only the values of k_{sig} that fall between the minimum and maximum values found by the band and slope constraints. For the Sakoe-Chiba local constraint with no band and slope constraints, all values of k_{sig} were included.

Step 3 – Create lookup table based on local continuity constraints

- For all the possible local transitions defined by the local continuity constraint, create a set of transitions indicating the number of transitions along the reference and signal index axes on the synchronisation grid.
- Assign a normalisation weight to each set of transitions based on equation 2.36.
- Store these values in a matrix of 3 columns. The first column contains the transitions in the reference index direction, the second column contains the transitions in the signal index direction and the third column contains the normalisation weights.

Step 4 – Local distance calculation

- For all the possible entries in the feasibility map, calculate the local Euclidean distance between the measured variables of the reference trajectory at index k_{ref} and the measured variables of the signal trajectory at index k_{sig} by equation 2.33.
- Store these local distances for later use.

Step 5 – Find cumulative distances

For every reference index k_{ref}:

- Create a vector $\boldsymbol{k_{sig}}$ filled with all the feasible signal indexes for k_{ref}. If $k_{ref} > 1$, concatenate with previous $\boldsymbol{k_{sig}}$ vector.
- Create a vector $\boldsymbol{k_{ref}}$, filled with the value of k_{ref} for each feasible signal index. If $k_{ref} > 1$, concatenate with the previous $\boldsymbol{k_{ref}}$ vector.
- Recall the feasible signal indexes corresponding to the reference index
- For every feasible signal index k_{sig}
    - Find all the allowable preceding coordinates on the synchronisation grid based on the local continuity constraints:
        i. Recall the number of transitions in the reference and signal index directions from the lookup table
        ii. Subtract the number of transitions in the reference index direction from k_{ref} to get the preceding reference index
        iii. Subtract the number of transitions in the signal index direction from k_{sig} to get the preceding signal index

iv. Create sets of preceding coordinates containing the preceding reference and signal indexes of each transition in the lookup table
v. Keep only the sets of preceding coordinates where both indexes in each set are greater than zero (i.e. they appear on the synchronisation grid)
vi. Keep only the sets of preceding coordinates that exist on the feasibility map (i.e. they fall within the global constraints)
vii. If no valid sets are retained (indicating the beginning of the path), set the value of the local and cumulated distances to zero
viii. Recall the normalisation weights of these valid preceding coordinates

- For each set of valid preceding coordinates
  i. Recall the corresponding stored cumulative distances
  ii. Recall the local distances between the variables of the reference and signal indexes
  iii. Multiply the local distance by its corresponding normalisation weight
  iv. Add the weighted local distance to the cumulative distance
- Find the set of preceding coordinates with the minimum cumulated distances
- Store the minimum cumulated distance, as well as the corresponding set of preceding coordinates in separate vectors, $\boldsymbol{D_{sig}}$ and $\boldsymbol{c(k_{prev})}$, respectively. These vectors are linked to the $\boldsymbol{k_{sig}}$ and $\boldsymbol{k_{ref}}$ vectors by their vector indexing (i.e. the $k\text{-}th$ value in the $\boldsymbol{D_{sig}}$ and $\boldsymbol{c(k_{prev})}$ vectors correspond with k_{sig} and k_{ref}, which are the $k\text{-}th$ values in their respective vectors).

- Normalise the cumulated distances using the normalisation factor in equation 2.36.

Step 6 – Find the optimal path

- Assign the first set of coordinates of the path as $(K_{\pi,ref}, K_{\pi,sig})$ to satisfy the final endpoint constraint
- Find the last set of preceding coordinates, $c(k_{prev})$, on the $\boldsymbol{c(k_{prev})}$ vector, as they correspond with $(K_{\pi,ref}, K_{\pi,sig})$
- Repeat until no preceding coordinates exist
  - Find the index of the coordinates of $c(k_{prev})$ using the $\boldsymbol{k_{sig}}$ and $\boldsymbol{k_{ref}}$ vectors
  - Assign the coordinates of $c(k_{prev})$ as the next set of coordinates in the optimal path

- Use the index found in the first step of this loop to find the next set of preceding coordinates and assign as the new $c(k_{prev})$

- Flip the optimal path from left to right so that the coordinates of the path read from beginning to end
- If the first set of coordinates on the path are not $(1,1)$, add these coordinates to the beginning of the path to satisfy the initial endpoint constraint
- For all k_{ref}
  - If any reference index k_{ref} from $1{:}\,K_{\pi,ref}$ does not appear on the optimal path, add it to the path directly after $(k_{ref} - 1)$
  - Find the corresponding value k_{sig} in the set of coordinates by linearly interpolating the corresponding signal indexes from the adjacent reference indexes and rounding to the nearest whole number

Step 6 – Asymmetric Synchronisation

1. For each k_{ref}:
   - Get the corresponding signal indexes k_{sig} from the optimal path. If there are duplicate entries for k_{ref} on the path, there will be more than one corresponding k_{sig} entry
   - Get the trajectories $\boldsymbol{x_{sig,k}} \in \mathbb{R}^{1\times J}$ for each k_{sig} from the slab scaled signal batch, $\boldsymbol{X_{sig}}$
   - Find the mean of these trajectories, $\overline{\boldsymbol{x}}_{\boldsymbol{sig,k}}$ for every variable J (if there is only one k_{ref} index, $\overline{\boldsymbol{x}}_{\boldsymbol{sig,k}} = \boldsymbol{x_{sig,k}}$
2. Store these values in a new matrix, $\boldsymbol{X_{warp}} \in \mathbb{R}^{K_{\pi,ref}\times J}$. The matrix contains the warped signal trajectories for every k_{ref} from $1{:}\,K_{\pi,ref}$
3. Remove the duplicate k_{ref} indexes from the optimal path and store as $\boldsymbol{F^*_{sig}} \in \mathbb{R}^{2\times K_{\pi,ref}}$. The corresponding k_{sig} index for duplicate k_{ref} indexes will be the first (lowest) k_{sig}

F.1.4 Multiphase Routine

Given: I batches of $\widetilde{\boldsymbol{X}}_{\boldsymbol{sync,i}} \in \mathbb{R}^{K_{ref}\times J}$

$\widetilde{\boldsymbol{Y}} \in \mathbb{R}^{I\times M}$

Initial number of retained components a_{ini}

Improvement threshold, T

minimum phase length, $minL$

k_y

Step 1: Batch-wise Unfolding

1. Concatenate all the batches $\widetilde{\boldsymbol{X}}_{\boldsymbol{sync,i}}$ such that the resulting matrix is of size $J \times (K_{ref} \times I)$
2. Reshape the matrix to size $I \times (JK_{ref})$. The resulting matrix $\boldsymbol{X_{BW}} \in \mathbb{R}^{I \times (JK_{ref})}$

Step 3 – Initialise PCs

Set a as a_{ini}

Step 3: – Cross Validation

For every i

1. Designate one row of $\widetilde{\boldsymbol{X}}_{\boldsymbol{BW}}$ as the test batch, $\widetilde{\boldsymbol{x}}_{\boldsymbol{test}} \in \mathbb{R}^{1 \times (JK_{ref})}$
2. Designate the corresponding row in $\widetilde{\boldsymbol{Y}}$ as $\widetilde{\boldsymbol{y}}_{\boldsymbol{test}} \in \mathbb{R}^{1 \times M}$
3. Designate the rest of $\widetilde{\boldsymbol{X}}_{\boldsymbol{BW}}$ as the training batch, $\widetilde{\boldsymbol{X}}_{\boldsymbol{train}} \in \mathbb{R}^{(I-1) \times (JK_{ref})}$
4. Designate the corresponding rows of $\widetilde{\boldsymbol{Y}}$ as $\widetilde{\boldsymbol{Y}}_{\boldsymbol{train}} \in \mathbb{R}^{(I-1) \times M}$
5. Auto scale $\widetilde{\boldsymbol{X}}_{\boldsymbol{train}}$ to get $\boldsymbol{X}_{\boldsymbol{train,auto}}$ and store the mean $\mu_{X,auto}$ and standard deviation $s_{X,auto}$
6. Auto scale $\widetilde{\boldsymbol{Y}}_{\boldsymbol{train}}$ to get $\boldsymbol{Y}_{\boldsymbol{train,auto}}$ and store the mean $\mu_{Y,auto}$ and standard deviation $s_{Y,auto}$
7. Subtract $\mu_{X,auto}$ from $\widetilde{\boldsymbol{x}}_{\boldsymbol{test}}$ and divide the result by $s_{X,auto}$ to get $\boldsymbol{x}_{\boldsymbol{test,auto}}$
8. Subtract $\mu_{Y,auto}$ from $\widetilde{\boldsymbol{y}}_{\boldsymbol{test}}$ and divide the result by $s_{Y,auto}$ to get $\boldsymbol{y}_{\boldsymbol{test,auto}}$
9. Use the SIMPLS algorithm with a components to find the regression matrix $\boldsymbol{\beta}_{\boldsymbol{train}} \in \mathbb{R}^{(JK_{ref}) \times (M-1)}$
10. for every k from $1{:}\,K$
    a. Create a matrix of zeroes of size of $\boldsymbol{X}_{\boldsymbol{train,auto}} \in \mathbb{R}^{(I-1) \times (JK_{ref})}$
    b. Assign the 1st to the Jk-th column of the matrix above as the 1st to the Jk-th columns of $\boldsymbol{X}_{\boldsymbol{train,auto}}$ to get $\boldsymbol{X}_{\boldsymbol{k,train,auto}}$
    c. multiply $\boldsymbol{X}_{\boldsymbol{k,train,auto}}$ by $\boldsymbol{\beta}_{\boldsymbol{train}}$ to get $\widehat{\boldsymbol{y}}_{\boldsymbol{k}}$

d. subtract $\widehat{\boldsymbol{y}}_{\boldsymbol{k}}$ from $\boldsymbol{y}_{\boldsymbol{test,auto}}$ and square the result

e. sum all the elements of the result above to get SSE_k

f. add SSE_k to the SSE

11. Divide SSE by K to get MSE_{cv}
12. Add MSE_{cv} to MSE

Divide MSE by I to get QPE

Step 4 – Modelling

1. Auto scale $\widetilde{\boldsymbol{X}}_{\boldsymbol{BW}}$ to get $\boldsymbol{X}_{\boldsymbol{BW,auto}}$
2. Auto scale $\widetilde{\boldsymbol{Y}}$ to get $\boldsymbol{Y}_{\boldsymbol{auto}}$
3. Use the SIMPLS algorithm with a components to find the regression matrix $\boldsymbol{\beta} \in \mathbb{R}^{(JK_{ref})\times M}$
4. for every k from $1{:}\,K_{ref}$
   a. Create a matrix of zeroes of size of $\boldsymbol{X}_{\boldsymbol{BW,auto}} \in \mathbb{R}^{I\times(JK_{ref})}$
   b. Assign the 1st to the Jk-th column of the matrix above as the 1st to the Jk-th columns of $\boldsymbol{X}_{\boldsymbol{BW,auto}}$ to get $\boldsymbol{X}_{\boldsymbol{k,BW,auto}}$
   c. multiply $\boldsymbol{X}_{\boldsymbol{k,BW,auto}}$ by $\boldsymbol{\beta}$ to get $\widehat{\boldsymbol{Y}}_{\boldsymbol{k}}$
   d. subtract $\widehat{\boldsymbol{Y}}_{\boldsymbol{k}}$ from $\boldsymbol{Y}_{\boldsymbol{auto}}$ and square the result
   e. sum all the elements of the result above to get SSE_k
   f. add SSE_k to the SSE
5. Divide SSE by K_{ref} to get mse
6. Add mse to MSE

Divide MSE by I to get QRE

Step 5 – Model Assignment

Store a, the $timespan = [1\ K]$, QPE, QRE and $\boldsymbol{\beta}$ **in** M

Step 6 – Add PC

1. recall QPE
2. get the data in $\widetilde{\boldsymbol{X}}_{\boldsymbol{BW}}$ from the $timespan(1) * J + 1{:}\,timespan(end) * J$
3. set $a = a + 1$
4. Execute Step 3 and get QPE

5. Subtract QPE from the QPE in M to get δ
6. Store a, the $timespan$, QPE, QRE and $\boldsymbol{\beta}$ **in** $NewM$

Step 7 – Split Data

1. Recall QPE and QRE from M
2. Recall $timespan$ from M
3. for k = $timespan(1) + minL - 1 : timespan(2) - minL$
   a. Auto scale $\widetilde{\boldsymbol{X}}_{\boldsymbol{BW}}$ to get $\boldsymbol{X}_{\boldsymbol{BW,auto}}$
   b. Auto scale $\widetilde{\boldsymbol{Y}}$ to get $\boldsymbol{Y}_{\boldsymbol{auto}}$
   c. Split $\boldsymbol{X}_{\boldsymbol{BW,auto}}$ into segments from $timespan(1) * J + 1 : k$ and $k * J : timespan(2)$ to get two segments $\boldsymbol{X}_{\boldsymbol{s1}}$ and $\boldsymbol{X}_{\boldsymbol{s2}}$
   d. for each segment
      i. execute 4 to get QRE_k
      ii. multiply QRE_k by the segment length
      iii. store in the vector $\boldsymbol{QRE}_{\boldsymbol{seg}} \in \mathbb{R}^{1 \times 2}$
   e. sum $\boldsymbol{QRE}_{\boldsymbol{seg}}$ and divide by $timespan(2) - timespan(1) + 1$
   f. subtract result from QRE to get the improvement threshold and stor as the *k*-th value in $\boldsymbol{\beta}_{\boldsymbol{k}}$
4. Find $optK$ that corresponds with the maximum value of $\boldsymbol{\beta}_{\boldsymbol{k}}$
   a. Split $\boldsymbol{X}_{\boldsymbol{BW,auto}}$ into segments from $timespan(1) * J + 1 : optK$ and $optK * J : timespan(2)$ to get two segments $\boldsymbol{X}_{\boldsymbol{s1}}$ and $\boldsymbol{X}_{\boldsymbol{s2}}$
   b. for each segment
      i. execute step 3 and 4 to get QPE_k and $\boldsymbol{\beta}_{\boldsymbol{seg}}$
      ii. multiply QPE_k by the segment length
      iii. store in the vector $\boldsymbol{QPE}_{\boldsymbol{seg}} \in \mathbb{R}^{1 \times 2}$
   c. Store a, the $timespan$, QPE_k, QRE_k and $\boldsymbol{\beta}_{\boldsymbol{seg}}$ **in** M_{seg}
   d. sum $\boldsymbol{QRE}_{\boldsymbol{seg}}$ and divide by $timespan(2) - timespan(1) + 1$
   e. subtract result from QRE recalled from M to get δ_s
   f. multiply δ_s by k_γ to get psi

Step 8 – Model Update

1. Compare psi from step 7 and δ from step 6

2. Compare largest result to improvement threshold T

3. If result $> T$, update with relevant model and repeat from step 5

4. Otherwise, return model

F.1 On-line Monitoring and Prediction

MPLS models trained off-line were applied to test data in an on-line fashion. Pre-processing and model information from the off-line training was prepared by the preliminary setup routine (section F.2.1).

A series of batch trajectories were used to simulate on-line monitoring and prediction. Each trajectory was processed individually, as would be the case in during on-line operation. At every time step k_{sig} from $1{:}\,K_{sig}$, the vector of process measurements at that time step, $\tilde{\boldsymbol{x}}_{\boldsymbol{k}} \in \mathbb{R}^{1\times J}$ was added to the raw data matrix $\tilde{\boldsymbol{X}}_{\boldsymbol{new}}$. Thereafter, the raw data matrix was pre-processed on-line using the pre-processing routine (section F.2.2). Models were then applied to the pre-processed data using the model application routine (section F.2.3).

F.2.1 Preliminary Setup Routine

Given: I batches of NOC data used for training, each of size $K_i \times J$

Stored: Π sets of reference trajectories, $\boldsymbol{X}_{\pi,\boldsymbol{ref}} \in \mathbb{R}^{K_{\pi,ref}\times J}$, for each operational phase π of the process

Corresponding set of Π phase boundary indexes for the reference trajectory

Π sets of slab scaling parameters, $\boldsymbol{\mu}_{\pi,\boldsymbol{slab}} \in \mathbb{R}^{1\times J}$ and $\boldsymbol{s}_{\pi,\boldsymbol{slab}} \in \mathbb{R}^{1\times J}$, for each operational phase π of the process

$\Pi \times I$ sets of optimal paths, $\boldsymbol{F^*}_{\boldsymbol{i},\pi} \in \mathbb{R}^{2\times K_{\pi,ref}}$, for each NOC batch

Π sets of variable weight matrixes, $\boldsymbol{W}_{\pi} \in \mathbb{R}^{J\times J}$

P sets of phase boundary indexes for each modelled phase p

P sets of auto scaling parameters, $\boldsymbol{\mu}_{\boldsymbol{p},\boldsymbol{auto}} \in \mathbb{R}^{K_{p,ref}\times J}$ and $\boldsymbol{s}_{\boldsymbol{p},\boldsymbol{auto}} \in \mathbb{R}^{K_{p,ref}\times J}$ for every modelled phase p

Size of window, γ

Step 1 – Reference Trajectory Preparation

1. For each phase π
   - Recall $\boldsymbol{X_{\pi,ref}}$, $\boldsymbol{\mu_{\pi,slab}}$ and $\boldsymbol{s_{\pi,slab}}$
   - Unscale $\boldsymbol{X_{\pi,ref}}$
     - At every reference batch time step k, multiply the standard deviation of each of the J columns with its corresponding variable ($x_{\pi,ref,jk}$) and add its corresponding mean:

$$\tilde{x}_{ref,jk} = \left(x_{\pi,ref,jk} \times s_{\pi,slab,j}\right) + \mu_{\pi,slab,j} \qquad \text{F.15}$$

     - Arrange the values in the matrix $\widetilde{\boldsymbol{X}}_{\pi,\boldsymbol{ref}} \in \mathbb{R}^{K_{\pi,ref} \times J}$
     - If $\pi = 1$, set $\widetilde{\boldsymbol{X}}_{\boldsymbol{ref}} = \widetilde{\boldsymbol{X}}_{\pi,\boldsymbol{ref}}$
     - If $\pi > 1$, concatenate $\widetilde{\boldsymbol{X}}_{\pi,\boldsymbol{ref}}$ with $\widetilde{\boldsymbol{X}}_{\boldsymbol{ref}}$
2. Find the slab scaling parameters of the NOC data for the entire trajectory
   - Concatenate all the batches such that the resulting matrix is of size $J \times \sum_{i=1}^{I} K_i$
   - Find the mean $\mu_{slab,j}$ for each of the J columns
   - Find the standard deviation $s_{slab,j}$ for each of the J columns. If the standard deviation of any of the columns is equal to zero, set equal to 1
   - Store each $\mu_{slab,j}$ and $s_{slab,j}$ for each variable j in $\boldsymbol{\mu}_{\boldsymbol{slab}} \in \mathbb{R}^{1 \times J}$ and $\boldsymbol{s}_{\boldsymbol{slab}} \in \mathbb{R}^{1 \times J}$ respectively
3. Slab scale the reference trajectory
   - At every time step k, subtract the mean of each of the J columns from its corresponding variable ($\tilde{x}_{ref,jk}$) and divide by its corresponding standard deviation:

$$x_{ref,jk} = \frac{\tilde{x}_{ref,jk} - \mu_{slab,j}}{s_{slab,j}} \qquad \text{F.16}$$

   - Store each $x_{ref,jk}$ in $\boldsymbol{X}_{ref} \in \mathbb{R}^{K_{ref} \times J}$

Step 2 – Synchronisation Parameters Preparation

1. Set and store the variable weights $\boldsymbol{W} = \boldsymbol{I} \in \mathbb{R}^{J \times J}$, where $\boldsymbol{I}$ is the identity matrix
2. Set the beginning coordinates of the final path of the new trajectory, $\boldsymbol{F^*}_{\boldsymbol{new}} \in \mathbb{R}^{2 \times 1}$ to $[1,1]$ and store
3. Set the starting window coordinates, $c(\gamma_{start})$ to $[1,1]$ and store

Step 3 – Synchronisation Search Space Limit Calculation

1. For each phase π
   - Find the optimal paths for the entire trajectory. For every NOC batch i
   - Recall $\boldsymbol{F^*_{i,\pi}}$
   - Let $\boldsymbol{F^*_i}$ be the optimal path for the i-th batch from $1{:}\,K_{ref}$
   - If $\pi = 1$, set $\boldsymbol{F^*_i = F^*_{i,\pi}}$
   - If $\pi > 1$
     - Find the current last set of coordinates of $\boldsymbol{F^*_i}$
     - Add this set to each set of coordinates in $\boldsymbol{F^*_{i,\pi}}$
     - Concatenate the result with $\boldsymbol{F^*_i}$

     The final $\boldsymbol{F^*_i} \in \mathbb{R}^{2\times K_{ref}}$ contains the optimal path for batch i over the entire trajectory. The first row contains the reference indexes k_{ref} and the second row contains the signal indexes k_{sig}
2. Find the lower and upper limits
   - Arrange the signal indexes of the paths in a matrix $\boldsymbol{F^*} \in \mathbb{R}^{I\times K_{ref}}$
   - Find the maximum and minimum of each column of $\boldsymbol{F^*}$
   - Store the calculated maximums and minimums in $\boldsymbol{u(F^*)} \in \mathbb{R}^{1\times K_{ref}}$ and $\boldsymbol{l(F^*)} \in \mathbb{R}^{1\times K_{ref}}$ respectively

Step 4 – Scaling Parameters Preparation

1. Concatenate $\boldsymbol{\mu_{p,auto}}$ and $\boldsymbol{s_{p,auto}}$ for every phase p such that $\boldsymbol{\mu_{auto}} \in \mathbb{R}^{K_{ref}\times J}$ and $\boldsymbol{s_{auto}} \in \mathbb{R}^{K_{ref}\times J}$
2. Store $\boldsymbol{\mu_{auto}}$ and $\boldsymbol{s_{auto}}$

Step 5 – Prediction Information Preparation

1. For every p, calculate the critical student's t-statistic at $\alpha = 0.05$ and 0.01 and $I\text{-}A\text{-}1$ degrees of freedom and store as $t_{I-A-1,\alpha/2}$

F.2.2 Pre-processing Routine

Given: A matrix of new data, $\boldsymbol{\tilde{X}_{new}} \in \mathbb{R}^{K_{new}\times J}$ up until the current time K_{new}

Stored: The reference trajectory, $\boldsymbol{X_{ref}} \in \mathbb{R}^{K_{ref}\times J}$

Slab scaling parameters $\boldsymbol{\mu_{slab}} \in \mathbb{R}^{1\times J}$ and $\boldsymbol{s_{slab}} \in \mathbb{R}^{1\times J}$

The variable weight matrix, $\boldsymbol{W} \in \mathbb{R}^{J \times J}$

The path limits $\boldsymbol{u}(\boldsymbol{F}^*) \in \mathbb{R}^{1 \times K_{ref}}$ and $\boldsymbol{l}(\boldsymbol{F}^*) \in \mathbb{R}^{1 \times K_{ref}}$

The final path $\boldsymbol{F}^*_{\boldsymbol{new}} \in \mathbb{R}^{2 \times (K_{new}-1)}$ up until the last synchronised path point

The starting coordinates of the window, $c(\gamma_{start})$

The warped trajectory up until the last synchronised path point, $\boldsymbol{X}_{new,sync} \in \mathbb{R}^{(K_{new}-1) \times J}$

P sets of phase boundary indexes for each modelled phase p

P sets of auto scaling parameters, $\boldsymbol{\mu}_{\boldsymbol{p,auto}} \in \mathbb{R}^{K_{p,ref} \times J}$ and $\boldsymbol{s}_{\boldsymbol{p,auto}} \in \mathbb{R}^{K_{p,ref} \times J}$ for every modelled phase p

Size of window, γ

Step 1 – Slab Scaling

1. Recall $\boldsymbol{\mu}_{\boldsymbol{slab}} \in \mathbb{R}^{1 \times J}$ and $\boldsymbol{s}_{\boldsymbol{slab}} \in \mathbb{R}^{1 \times J}$
2. For each k from $1{:}\, K_{new}$
    - Isolate the k-th row of $\widetilde{\boldsymbol{X}}_{\boldsymbol{new}}$ to get $\widetilde{\boldsymbol{x}}_{\boldsymbol{new,k}} \in \mathbb{R}^{1 \times J}$
    - Subtract the *j*-th element of $\boldsymbol{\mu}_{\boldsymbol{slab}}$ from the *j*-th element of $\widetilde{\boldsymbol{x}}_{\boldsymbol{new,k}}$
    - Divide the *j*-th of the resulting vector by the *j*-th element of $\boldsymbol{s}_{\boldsymbol{slab}}$
    - Store the resulting vector as $\boldsymbol{X}_{\boldsymbol{new,slab}} \in \mathbb{R}^{K_{new} \times J}$

Step 2 – Synchronisation

1. Transpose $\boldsymbol{X}_{\boldsymbol{new,slab}}$ such that $\boldsymbol{X}^{\boldsymbol{T}}_{\boldsymbol{new,slab}} \in \mathbb{R}^{J \times K_{new}}$
2. Invert the path limits $\boldsymbol{u}(\boldsymbol{F}^*) \in \mathbb{R}^{1 \times K_{ref}}$ and $\boldsymbol{l}(\boldsymbol{F}^*) \in \mathbb{R}^{1 \times K_{ref}}$ to find $\boldsymbol{l}(\boldsymbol{F}^*)_{\boldsymbol{sig}} \in \mathbb{R}^{1 \times K_{new}}$ and $\boldsymbol{u}(\boldsymbol{F}^*)_{\boldsymbol{sig}} \in \mathbb{R}^{1 \times K_{new}}$
    - Find all the unique signal indexes that appear in $\boldsymbol{u}(\boldsymbol{F}^*)$
    - For each unique signal index k_{new} from $1{:}\, \max(\boldsymbol{u}(\boldsymbol{F}^*))$
        - Find the smallest index of the $\boldsymbol{u}(\boldsymbol{F}^*)$ vector corresponding to k_{new}
        - Store the value of the index in the k_{new}-th index of the $\boldsymbol{l}(\boldsymbol{F}^*)_{\boldsymbol{sig}}$ vector
    - For every other k_{new} from $1{:}\, \max(\boldsymbol{u}(\boldsymbol{F}^*))$
        - Find the index of the smallest value in $\boldsymbol{u}(\boldsymbol{F}^*)$ larger than k_{new}

- Store the value of this index in the k_{new}-th index of the $\boldsymbol{l}(\boldsymbol{F}^*)_{\boldsymbol{sig}}$ vector

- Find all the unique signal indexes that appear in $\boldsymbol{l}(\boldsymbol{F}^*)$
- For each unique signal index k_{new} from $1{:}\max(\boldsymbol{u}(\boldsymbol{F}^*))$
  - Find the largest index of the $\boldsymbol{l}(\boldsymbol{F}^*)$ vector corresponding to k_{new}
  - Store the value of the index in the k_{new}-th index of the $\boldsymbol{u}(\boldsymbol{F}^*)_{\boldsymbol{sig}}$ vector
- For every other k_{new} from $1{:}\max(\boldsymbol{u}(\boldsymbol{F}^*))$
  - Find the index of the largest value in $\boldsymbol{l}(\boldsymbol{F}^*)$ smaller than k_{new}
  - Store the value of this index in the k_{new}-th index of the $\boldsymbol{u}(\boldsymbol{F}^*)_{\boldsymbol{sig}}$ vector

3. Recall the window length, γ
4. Create a set of coordinates to define the vertices of the warping window
   - Find the value of the k_{new}-th index on $\boldsymbol{u}(\boldsymbol{F}^*)_{\boldsymbol{sig}}$ to get $k_{ref,upper}$ and save the coordinates as $c(k_{new,upper})$
   - Find the value of the k_{new}-th index on $\boldsymbol{l}(\boldsymbol{F}^*)_{\boldsymbol{sig}}$ to get $k_{ref,lower}$ and save the coordinates as $c(k_{new,lower})$
   - Recall the starting coordinates of the window, $c(\gamma_{start})$
   - If the reference index of $c(\gamma_{start})$ is larger than $k_{ref,lower}$, overwrite $k_{ref,lower}$ with this reference index to satisfy the local continuity constraints
   - If $k_{new} > \gamma$, find the starting signal index, $k_{new-\gamma+1} = k_{new} - \gamma + 1$
   - If $k_{new} \leq \gamma$, set $k_{new-\gamma+1} = 1$
   - Find the value of the $k_{new-\gamma+1}$-th index on $\boldsymbol{u}(\boldsymbol{F}^*)_{\boldsymbol{sig}}$ to get $k_{ref,upper,start}$ and save the coordinates as $c(k_{new-\gamma+1,upper})$
   - Put $c(\gamma_{start})$, $c(k_{new-\gamma+1,upper})$, $c(k_{new,lower})$ and $c(k_{new,upper})$ in the vector $\boldsymbol{\gamma} \in \mathbb{R}^{2\times 4}$
5. Isolate the values on $\boldsymbol{u}(\boldsymbol{F}^*)$ and $\boldsymbol{l}(\boldsymbol{F}^*)$ from $k_{ref,\gamma_{start}}$ to $k_{ref,upper}$ to get $\boldsymbol{u}_{\gamma}(\boldsymbol{F}^*)$ and $\boldsymbol{l}_{\gamma}(\boldsymbol{F}^*)$
6. Reset the window coordinate indexes
   - Subtract $c(\gamma_{start})$ from each set of coordinates in $\boldsymbol{\gamma}$ and add $[1,1]$
   - Subtract $c(\gamma_{start})$ from each set of coordinates in $\boldsymbol{u}_{\gamma}(\boldsymbol{F}^*)$ and $\boldsymbol{l}_{\gamma}(\boldsymbol{F}^*)$ and add $[1,1]$
7. Create a feasibility map

   For every reference index n_{ref} within the window for $1{:}\,n_{ref,upper}$

- Find the range of feasible signal indexes between the lower and upper limits, $l_\gamma(n_{ref}):u_\gamma(n_{ref})$
- Exclude values that fall outside the signal index range $(1:\gamma)$

8. Create a lookup table based on the local continuity constraints
    - For all the possible local transitions defined by the local continuity constraint, create a set of transitions indicating the number of transitions along the reference and signal index axes on the synchronisation grid.
    - Assign a normalisation weight to each set of transitions based on equation 2.35.
    - Store these values in a matrix of 3 columns. The first column contains the transitions in the reference index direction, the second column contains the transitions in the signal index direction and the third column contains the normalisation weights.
9. Recall the reference trajectory, $\boldsymbol{X}_{ref}$ and the weighting matrix, $\boldsymbol{W}$
10. Local Distance Calculation
    - For all the possible entries in the feasibility map
        - Find the actual index of $k_{ref} = n_{ref} + k_{ref,\gamma_{start}} - 1$
        - Find the actual index of $k_{new} = n_{new} + k_{new,\gamma_{start}} - 1$
        - Calculate the local Euclidean distance between the measured variables of the reference trajectory at index k_{ref} and the measured variables of the signal trajectory at index k_{new} by equation 2.33.
    - Store these local distances for later use using their window indexes, n_{ref} and n_{new}
11. Find cumulative distances

    For every reference index n_{ref}:

    - Create a vector $\boldsymbol{n_{new}}$filled with all the feasible signal indexes for n_{ref}. If $n_{ref} > 1$, concatenate with previous $\boldsymbol{n_{ref}}$ vector.
    - Create a vector $\boldsymbol{n_{ref}}$, filled with the value of n_{ref} for each feasible signal index. If $n_{ref} > 1$, concatenate with the previous $\boldsymbol{n_{ref}}$ vector.
    - Recall the feasible signal indexes corresponding to the reference index
    - For every feasible signal index n_{ref}
        - Find all the allowable preceding coordinates on the synchronisation grid based on the local continuity constraints:

        - Recall the number of transitions in the reference and signal index directions from the lookup table
        - Subtract the number of transitions in the reference index direction from n_{ref} to get the preceding reference index
        - Subtract the number of transitions in the signal index direction from n_{new} to get the preceding signal index
        - Create sets of preceding coordinates containing the preceding reference and signal indexes of each transition in the lookup table
        - Keep only the sets of preceding coordinates where both indexes in each set are greater than zero (i.e. they appear on the synchronisation grid)
        - Keep only the sets of preceding coordinates that exist on the feasibility map (i.e. they fall within the global constraints)
        - If no valid sets are retained (indicating the beginning of the path), set the value of the local and cumulated distances to zero
        - Recall the normalisation weights of these valid preceding coordinates
    - For each set of valid preceding coordinates
        - Recall the corresponding stored cumulative distances
        - Recall the local distances between the variables of the reference and signal indexes
        - Multiply the local distance by its corresponding normalisation weight
        - Add the weighted local distance to the cumulative distance
    - Find the set of preceding coordinates with the minimum cumulated distances
    - Store the minimum cumulated distance, as well as the corresponding set of preceding coordinates in separate vectors, $\boldsymbol{D_{new}}$ and $\boldsymbol{c(n_{prev})}$, respectively. These vectors are linked to the $\boldsymbol{n_{new}}$ and $\boldsymbol{n_{ref}}$ vectors by their vector indexing (i.e. the $\boldsymbol{n\text{-}th}$ value in the $\boldsymbol{D_{new}}$ and $\boldsymbol{c(n_{prev})}$ vectors correspond with n_{new} and n_{ref}, which are the $\boldsymbol{n\text{-}th}$ values in their respective vectors).
- Normalise the cumulated distances using the normalisation factor in equation 2.36.

12. Find the Optimal Path within the window, $\boldsymbol{f^*_{\gamma}}$
    - Find all the values in $\boldsymbol{D_{new}}$ that correspond with $n_{new} = \gamma$. The resulting vector $\boldsymbol{D_{new,end}}$ should have a value corresponding to every feasible reference endpoint, $n_{ref,lower}{:}\, n_{ref,upper}$
    - For every $n_{ref} = n_{ref,lower}{:}\, n_{ref,upper}$
        - Add the weighted local distance corresponding between $n_{new} = \gamma$ and n_{ref} to the corresponding cumulated distance on $\boldsymbol{D_{new,end}}$
        - Normalise the distance by dividing by $n_{new} + n_{ref}$
    - Find the minimum distance on $\boldsymbol{D_{new,end}}$ and locate its corresponding set of preceding coordinates, $c(n_{prev})$, on the $\boldsymbol{c(n_{prev})}$ vector
    - Repeat until no preceding coordinates exist
        - Find the index of the coordinates of $c(n_{prev})$ using the $\boldsymbol{n_{new}}$ and $\boldsymbol{n_{ref}}$ vectors
        - Assign the coordinates of $c(n_{prev})$ as the next set of coordinates in the optimal path
        - Use the index found in the first step of this loop to find the next set of preceding coordinates and assign as the new $c(n_{prev})$
    - Flip the optimal path from left to right so that the coordinates of the path read from beginning to end
    - If the first set of coordinates on the path are not $[1,1]$, add these coordinates to the beginning of the path to satisfy the initial endpoint constraint
    - For all n_{ref}
        - If any reference index n_{ref} from 1 until the optimal reference index, $n_{ref,opt}$, does not appear on the optimal path, add it to the path directly after $(n_{ref} - 1)$
        - Find the corresponding value n_{new} in the set of coordinates by linearly interpolating the corresponding signal indexes from the adjacent reference indexes and rounding to the nearest whole number
13. Add the starting window coordinates and subtract 1 to the optimal window path to get the actual indices of the optimal path, $\boldsymbol{F^*_{\gamma}} = \boldsymbol{f^*_{\gamma}} + c(\gamma_{start})$
14. Asymmetric Synchronisation

For each k_{ref}:

- Get the corresponding signal indexes k_{new} from the optimal path. If there are duplicate entries for k_{ref} on the path, there will be more than one corresponding k_{new} entry
- Get the trajectories $\boldsymbol{x_{new,k}} \in \mathbb{R}^{1\times J}$ for each k_{new} from the slab scaled signal batch, $\boldsymbol{X_{new}}$
- Find the mean of these trajectories, $\bar{\boldsymbol{x}}_{\boldsymbol{new,k}}$ for every variable j (if there is only one k_{ref} index, $\bar{\boldsymbol{x}}_{\boldsymbol{new,k}} = \boldsymbol{x_{new,k}}$

15. Store these values in a new matrix, $\boldsymbol{X_{\gamma,sync,}}$. The matrix contains the warped signal trajectories for every k_{ref} from $n_{ref,\gamma_{start}}: n_{ref,opt}$
16. Remove the duplicate k_{ref} indexes from the optimal path and store in $\boldsymbol{F^*}_{\gamma}$. The corresponding k_{new} index for duplicate k_{ref} indexes will be the first (lowest) k_{new}
17. Add $\boldsymbol{X_{\gamma,sync}}$ to the warped matrix, $\boldsymbol{X}_{new,sync}$ at every reference index on $\boldsymbol{F^*}_{\gamma}$
18. Invert $\boldsymbol{F^*}_{\gamma}$ to form $\boldsymbol{F^*}_{\gamma,new}$ such that there is a corresponding reference index for every signal index. If a signal index k_{new} appears in the window but not on $\boldsymbol{F^*}_{\gamma}$, use the largest reference index smaller than k_{new} as its corresponding k_{ref} on $\boldsymbol{F^*}_{\gamma,new}$
19. Add $\boldsymbol{F^*}_{\gamma,new}$ to the optimal path of the entire trajectory, $\boldsymbol{F^*_{new}}$, at each signal point within the window
20. Update the limits
    - If the last reference index on $\boldsymbol{F^*_{new}}$ is equal to the K_{new}-th point on $\boldsymbol{l}(\boldsymbol{F^*})_{\boldsymbol{sig}}$, add 2 to every point on $\boldsymbol{l}(\boldsymbol{F^*})_{\boldsymbol{sig}}$ from the K_{new}-th point until the end of the vector
    - If the last reference index on $\boldsymbol{F^*_{new}}$ is equal to the K_{new}-th point on $\boldsymbol{u}(\boldsymbol{F^*})_{\boldsymbol{sig}}$
        - Assign this reference index, $k_{ref,opt}$ to the $(K_{new}+1)$-th and $(K_{new}+2)$-th points
        - Subtract $k_{ref,opt}$ from the $(K_{new}+2)$-th point, and then subtract this value from every value on $\boldsymbol{u}(\boldsymbol{F^*})_{\boldsymbol{sig}}$ from $(K_{new}+3)$ to the end of the vector
21. Invert the limits
    - Find all the unique signal indexes that appear in $\boldsymbol{u}(\boldsymbol{F^*})_{\boldsymbol{sig}}$
    - For each unique reference index k_{ref} from $1: K_{ref}$
        - Find the smallest index of the $\boldsymbol{u}(\boldsymbol{F^*})_{\boldsymbol{sig}}$ vector corresponding to k_{ref}

- Store the value of the index in the k_{ref}-th index of the $\boldsymbol{l}(\boldsymbol{F}^*)$ vector

- For every other k_{new} from $1{:}\,K_{ref}$
  - Find the index of the smallest value in $\boldsymbol{u}(\boldsymbol{F}^*)_{\boldsymbol{sig}}$ larger than k_{new}
  - Store the value of this index in the k_{new}-th index of the $\boldsymbol{l}(\boldsymbol{F}^*)$ vector
- Find all the unique reference indexes that appear in $\boldsymbol{l}(\boldsymbol{F}^*)_{\boldsymbol{sig}}$
- For each unique signal index k_{ref} from $1{:}\,K_{ref}$
  - Find the largest index of the $\boldsymbol{l}(\boldsymbol{F}^*)_{\boldsymbol{sig}}$ vector corresponding to k_{ref}
  - Store the value of the index in the k_{ref}-th index of the $\boldsymbol{u}(\boldsymbol{F}^*)$ vector
- For every other k_{ref} from $1{:}\,K_{ref}$
  - Find the index of the largest value in $\boldsymbol{l}(\boldsymbol{F}^*)_{\boldsymbol{sig}}$ smaller than k_{new}
  - Store the value of this index in the k_{ref}-th index of the $\boldsymbol{u}(\boldsymbol{F}^*)$ vector

22. Set the starting coordinates $c(\gamma_{start})$ of the window as the index on the optimal path corresponding with $K_{new} - \gamma + 2$ (or 1 if $K_{new} < \gamma - 1$)

Step 3 – Phase Identification

1. Recall the phase boundary indexes
2. Find the *p*-th set of phase boundary indexes that the final reference point, $k_{ref,opt}$ of the optimal path falls between
3. Store *p*

Step 4 – Unscaling

1. Recall $\boldsymbol{\mu}_{\boldsymbol{slab}} \in \mathbb{R}^{1\times J}$ and $\boldsymbol{s}_{\boldsymbol{slab}} \in \mathbb{R}^{1\times J}$
2. For each k from $1{:}\,k_{ref,opt}$
   - Isolate the k-th row of $\boldsymbol{X}_{\boldsymbol{new,sync}}$ to get $\boldsymbol{x}_{\boldsymbol{new,sync}} \in \mathbb{R}^{1\times J}$
   - Multiply the *j*-th of $\boldsymbol{x}_{\boldsymbol{new,slab}}$ with the *j*-th element of $\boldsymbol{s}_{\boldsymbol{slab}}$
   - Add the *j*-th element of $\boldsymbol{\mu}_{\boldsymbol{slab}}$ to the *j*-th element of the resulting vector
   - Store the resulting vector as $\widetilde{\boldsymbol{X}}_{\boldsymbol{new,sync}} \in \mathbb{R}^{K_{p,ref}\times J}$

Step 5 – Auto Scaling

1. Recall $\boldsymbol{\mu}_{\boldsymbol{auto}} \in \mathbb{R}^{K_{ref}\times J}$ and $\boldsymbol{s}_{\boldsymbol{auto}} \in \mathbb{R}^{K_{ref}\times J}$
2. Isolate the first $k_{ref,opt}$ rows of $\boldsymbol{\mu}_{\boldsymbol{auto}}$ and $\boldsymbol{s}_{\boldsymbol{auto}}$
3. Subtract the element in the *k*-th row and *j*-th column of $\boldsymbol{\mu}_{\boldsymbol{auto}}$ from corresponding element in $\widetilde{\boldsymbol{X}}_{\boldsymbol{new,sync}}$

4. Divide element in the *k*-th row and *j*-th column of the resulting vector by the corresponding element of $\boldsymbol{s_{slab}}$
5. Store the resulting vector as $\boldsymbol{X_{new,auto}} \in \mathbb{R}^{k_{ref,opt} \times J}$

Step 6 – Phase Data Isolation

1. Add zeroes to the matrix $\boldsymbol{X_{new,auto}}$ from the $k_{ref,opt}$-th to the K_{ref}-th row such that $\boldsymbol{X_{new,auto}} \in \mathbb{R}^{K_{ref} \times J}$
2. Isolate the rows of data from the indexes of the starting phase boundary to end phase boundary of the stored phase *p* such that $\boldsymbol{X_{p,new}} \in \mathbb{R}^{K_{p,ref} \times J}$

Step 7 – Batch-wise Unfolding

1. Reshape the matrix to size $1 \times (JK_{p,ref})$. The resulting matrix $\boldsymbol{X_{p,new,BW}} \in \mathbb{R}^{1 \times (JK_{p,ref})}$

F.2.3 Model Application Routine

Given: A matrix of synchronised, auto scaled and unfolded data, $\boldsymbol{X_{p,new,unfold}} \in \mathbb{R}^{K_{p,ref} \times J}$ for the current phase *p*

Stored: *P* sets of modelling results, $\boldsymbol{P_p} \in \mathbb{R}^{A_p \times (JK_{p,ref})}$, $\boldsymbol{T_p} \in \mathbb{R}^{I \times A_p}$, $\boldsymbol{\beta_p} \in \mathbb{R}^{(JK_{p,ref}) \times M}$ and $\boldsymbol{R_p} \in \mathbb{R}^{(JK_{p,ref}) \times A_p}$ for each modelled phase *p*

P sets of monitoring parameters, c_p and $\boldsymbol{S_p} \in \mathbb{R}^{K_{p,ref} \times A}$ for each modelled phase *p*

The Mean Squared Error of the model, $MSE_{y,p}$

The Student's *t*-statistic, $t_{I-A-1,\alpha/2}$

Step 1 – Predict End-of-Batch Quality

1. Recall $\boldsymbol{\beta_p}$
2. Calculate $\hat{\boldsymbol{y}}_{\boldsymbol{p,new}}$ and store

$$\hat{\boldsymbol{y}}_{\boldsymbol{p,new}} = \boldsymbol{X_{p,new,BW}}\, \boldsymbol{\beta_p} \qquad \text{F.17}$$

Step 2 – Calculate End-of-Batch Confidence Limits

1. Recall $\boldsymbol{T_p}$, $\boldsymbol{R_p}$, $t_{I-A-1,\alpha/2}$ and $MSE_{y,p}$
2. Calculate $\hat{\boldsymbol{t}}_{\boldsymbol{p,new}}$

$$\hat{\boldsymbol{t}}_{\boldsymbol{p,new}} = \boldsymbol{X}_{\boldsymbol{p,new,BW}}\, \boldsymbol{R}_{\boldsymbol{p}} \qquad \text{F.18}$$

3. Find the confidence limits using equation 2.28 and store

Step 3 – Calculate Hotellings T^2 Statistic

1. Recall c_p and $\boldsymbol{S_p}$
2. Isolate the first $Jk_{p,ref,opt}$ rows of $\boldsymbol{R_p}$ to create $\boldsymbol{R_{p,k}} \in \mathbb{R}^{(Jk_{p,ref,opt}) \times A_p}$
3. Pad the $(Jk_{p,ref,opt} + 1)$ -th to the $(JK_{p,ref})$ rows with zeros such that $\boldsymbol{R_{p,k}} \in \mathbb{R}^{(JK_{p,ref}) \times A_p}$
4. Calculate $\hat{\boldsymbol{t}}_{\boldsymbol{p,k}}$

$$\hat{\boldsymbol{t}}_{\boldsymbol{p,k}} = \boldsymbol{X}_{\boldsymbol{p,new,BW}}\, \boldsymbol{R}_{\boldsymbol{p,k}} \qquad \text{F.19}$$

5. Isolate the $(k_{p,ref,opt})$-th row of $\boldsymbol{S_p}$ to get $\boldsymbol{s_{p,k}}$
6. Divide the a-th element of $\hat{\boldsymbol{t}}_{\boldsymbol{p,k}}$ by the a-th element of $\boldsymbol{s_{p,k}}$
7. Multiply the result by c_p to get the Hotelling's T^2 statistic, ${T_{A_p}}^2$
8. Store ${T_{A_p}}^2$

Step 4 – Calculate SPE_x Statistic

1. Recall $\boldsymbol{P_p}$
2. Calculate $\hat{\boldsymbol{X}}_{\boldsymbol{p,k}}$

$$\hat{\boldsymbol{X}}_{\boldsymbol{p,k}} = \hat{\boldsymbol{t}}_{\boldsymbol{p,k}}\, \boldsymbol{P_p} \qquad \text{F.20}$$

3. Isolate the $(k_{p,ref,opt})$-th row of $\boldsymbol{X_{p,k}}$ and $\hat{\boldsymbol{X}}_{\boldsymbol{p,k}}$ to get $\boldsymbol{x_{p,k}}$ and $\hat{\boldsymbol{x}}_{\boldsymbol{p,k}}$
4. Subtract $\hat{\boldsymbol{x}}_{\boldsymbol{p,k}}$ from $\boldsymbol{x_{p,k}}$ and square each element of the resulting vector to get $\boldsymbol{SPE_{x,p,k}} \in \mathbb{R}^{1 \times J}$
5. Calculate the sum of all the elements in $\boldsymbol{SPE_{x,p,k}}$ to get $SPE_{x,p,k}$
6. Store $SPE_{x,p,k}$

F.3 On the Weighting of Variables

Kassidas et al. (1998) proposed an iterative method to determine the values of the diagonals in the $\boldsymbol{W}$ matrix. This method was criticised by Ramaker et al. (2003), who proposed an alternative approach. González-Martínez et al. (2011) used the geometric mean of these weights in their analysis so as to retain the features of both. The weighting matrix may have a significant effect on the way the same signal is warped to the reference trajectory, and may even have an effect on the choice of reference trajectory if the variable weighting procedure is done using the temporary reference trajectory (i.e. before the actual reference trajectory is chosen).

In the pre-processing routine above, each phase was separated based on known phase boundaries and synchronised separately. This was done in order to reduce the size of the synchronisation grid and subsequently the computational time of the synchronisation. Incorporating the methods discuss in the previous paragraph to calculate the weighting matrix could result in different weighting matrices for each operational phase of the process. This provides a complication in on-line monitoring when the current phase is only identified after synchronisation has been done on the batch trajectory. If different weighting matrices are used for the different operational phases in off-line synchronisation, similarly these matrices would have to be applied to the correct operational phase in on-line synchronisation. This is not particularly feasible; thus the more sensible approach would be to apply one weighting matrix across the entire batch trajectory. Finding one weighting matrix using the iterative approaches would require the entire trajectory to be synchronised in the pre-processing step as opposed to synchronising each phase individually. However, the computational expense for such a procedure is significant. Additionally, process variables may behave differently in different phases, and calculating the weighting matrix across the entire batch trajectory may not provide much improved synchronisation accuracy. Thus, the pragmatic decision was made not to calculate the weights iteratively. Instead, each variable was weighted equally by setting the weighting matrix to a diagonal matrix of ones i.e. $\boldsymbol{W} = \boldsymbol{I} \in \mathbb{R}^{J \times J}$. This also provides a level of objectivity to the comparison of the off-line versus on-line synchronisation techniques as the weighting matrix would not have any influence on the warping of the trajectories. However, it is recommended for future work to study the effect of weighting on the synchronisation accuracy of off-line and on-line synchronisation techniques.